前　言

数值计算的复杂性理论，在 20 世纪 80 年代取得了若干重要进展，引起了人们的注意. 这一进展既发生在应用数学领域，又植根于纯粹数学的深层，令两方面的数学家都十分关心.

本书力图以浅显易懂的语言，准确地向读者介绍数值计算复杂性理论的重要进展. 斯梅尔教授的开创性工作，是全书的重点.

第一章以易学易做的不动点迭代为引子，说明收敛性（算法是否成功）和复杂性（计算效率如何）的关系，着重阐明为什么“多项式时间算法”那么重要. 这一章，是基本概念的准备. 第二章专写中学生可以完全看懂的库恩算法及其复杂性讨论的结果. 这一章多写一些完全值得，因为这是当代前沿研究中罕见的可以讲得相当浅白的工作. 第三章阐述斯梅尔关于牛顿算法的复杂性研究的开创性工作，以及他的工作对复杂性理论的深远影响. 第四章介绍线性规划问题的意义，算法研究和复杂性讨论两者相得益彰的发展，这是十年来复杂性讨论中最引人注目的论题. 这一章的最后，还简单谈谈复杂性讨论的学科环境：应用问题向几何学、拓扑学、代数几何、积分几何与

VISIT…

LANZAROTE Caliente.com

几何概率、单叶函数理论等方面寻求工具.

这么重大的进展，如此丰富的内容，不可能在这样一本小册子中完整论述. 笔者牢记为具有中学数学基础的数学爱好者写作的宗旨，务必保证他们可以看懂大部分内容，在有些地方精心制作粗线条的描述，不失为一种好的处理. 实际上，这对作者也是一次考试. 但我们明确重在阐明和揭示数学思想，这是数学创造的真谛. 一些若仔细论证则难免繁难的地方，就代之以粗线条的和力求准确的定性说明. 书中还穿插若干有趣的科学故事，读者定会从中得到启迪.

作为一本高层次的科普著作，我们不准备采用专著的撰写形式. 除了正文的写作将具有上一段明言的特点以外，我们也不详列参考文献和名词索引. 对于希望深入了解数值计算复杂性理论的进展和内容的读者，我们仅列以下中文书籍：中山大学出版社 1986 年出版的《单纯不动点算法基础》，科学出版社 1989 年出版的《代数方程组与计算复杂性理论》，重庆出版社 1990 年出版的《同伦方法引论》，以及国防工业出版社 1991 年出版的《拓扑理论及其应用》. 这些著作中指出了重要的原始文献. 但是必须再一次强调，以上著作只是进一步专门阅读的建议，而不是对读者的要求. 本书除了希望读者具有良好的创造思维以外，只要求读者具备运用中学数学进行思考的能力.

最后我们指出，本书与计算机科学的复杂性理论不同，完全不谈可计算性、二进制、图灵机和人工智能. 具体地说，我们假定读者从未接触过电子计算机. 当然，计算机科学的复杂性理论和本书所写的数值计算的复杂性理论，二者有深刻的联系，我们设想在条件成熟的时候，深刻的联系会在一些读者的头脑中油然自生.

“走向数学”丛书的组织与出版，是具有深远影响的工程.

在中学数学的基础上，用现代观点向高中生，中学教师、大学低年级学生，工程技术人员和数学爱好者介绍数学的一些创造思想，使大家真正地认识数学，了解数学，热爱数学，走向数学，是一种很有意义的工作.

在师长和同道的鼓励和鞭策之下，笔者承担了本书的写作任务. 笔者学识陋浅，虽兢兢业业写作，谬误亦恐难免，诚望读者批评指教，不负数学为用之美.

江泽涵教授从特例入手的研究方法，吴文俊教授对构造性数学和计算机的兴趣，都很使笔者受益. 谨以此书的写作，献给两位敬爱的老师.

王则柯谨识

1990 年夏于中山大学岭南学院

目　录

前言（王元）…………………………………………………… 1
前言（王则柯）…………………………………………………… 3

第一章　数值计算的复杂性问题…………………………… 1
§1　代数方程的不动点迭代算法 ……………………… 2
§2　收敛性和复杂性——算法优劣判别的两个层次
…………………………………………………… 10
§3　可怕的指数增长——古印度数学故事…………… 14
§4　寻求多项式时间算法………………………………… 20
§5　温故而创新的代数基本定理………………………… 24
第二章　库恩算法及其计算复杂性 ………………………… 29
§1　库恩算法的描述……………………………………… 30
§2　可行性和收敛性的论证……………………………… 37
§3　全标三角形与根的距离……………………………… 42
§4　积木结构的计算复杂性讨论………………………… 46
第三章　斯梅尔对牛顿算法的研究 ………………………… 52
§1　多项式求根的牛顿算法……………………………… 53
§2　牛顿方法什么时候听话……………………………… 58
§3　概率论定牛顿算法是多项式时间算法…………… 64
§4　从最坏情形分析到概率情形分析………………… 70
§5　算法之比较和配合…………………………………… 74

第四章　线性规划问题算法的竞争 ………………………… 79
§1　线性规划问题……………………………………………… 80
§2　丹齐克的单纯形算法…………………………………… 89
§3　哈奇安的椭球算法……………………………………… 94
§4　卡马卡的内点算法……………………………………… 99
§5　斯梅尔论证了丹齐克的信念 ………………………… 102
§6　复杂性讨论的学科环境 ……………………………… 106

编后记（冯克勤）………………………………………………… 110

第一章　数值计算的复杂性问题

复杂性(complexity)作为一个科学名词出现，是最近20年的事. 这也是在《数学百科辞典》和《简明不列颠百科全书》中找不到"复杂性"条目的原因.

复杂性的科学研究，有漫延的趋势. 例如，新近已经有人讨论碎形几何（The geometry of fractal sets）集合的复杂性问题. 不过，已经成大气候的却还只限于两个方面：计算机科学的复杂性理论和数值计算的复杂性理论.

读者听说过二进制数、图灵机、递归函数和人工智能，这些内容都属于理论计算机科学复杂性讨论的范围. 粗略地说，计算机科学的复杂性讨论，是有关机器（指计算机）原理、功能和机器设计的讨论. 数值计算的复杂性讨论，则主要是数值计算方法的效率和数值计算方法的设计的讨论. 当然，两种复杂性讨论之间存在着深刻的联系，它们都是电子数字计算机蓬勃发展和广泛应用的产物. 但是，它们之间又有明确的界限. 例如，为了研究计算方法的复杂性问题，你不必知道多少机器的原理，但是要研究计算机科学的复杂性问题，非得精通机器的

原理不行．现代的数值计算，当然离不开计算机．但是，使用机器和了解机器的原理，毕竟是两回事．现在计算器已经很普及，人人都会用，然而这些人多半不知道也不关心它的运行原理．在这个意义上，可以说数值计算的复杂性讨论，研究的是如何更好地利用机器进行数值计算的问题．

本书的宗旨，是在中学数学的基础上，向读者介绍数值计算复杂性讨论的思想和进展．复杂性讨论的应用性很强，但是读者将会看到，纯粹数学中原先被认为是很抽象的一些理论，在这个应用科学的研究中表现出强大的生命力．

我们从最简单的例子开始，深入浅出地叙述数值计算复杂性讨论的思想和进展．个别细述难免繁难的地方，则代之以粗线条的定性说明．文武之道，一张一弛．书中还穿插若干有趣的科学故事．

§1　代数方程的不动点迭代算法

为了说明数值计算的复杂性问题，我们先介绍一种特殊的计算方法——代数方程的不动点迭代方法．这种方法一学就会．

中学数学讨论得最多的代数方程是形如

$$ax^2+bx+c=0,\ a\neq 0$$

这样的一元二次方程，它的两个根可以按照

$$x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

这个求根公式算出来．这种把代数方程的根用方程系数经有限次加、减、乘、除和开方运算表示出来的方法，叫做代数方程的**代数解法**．

爱好数学的读者知道，三次方程和四次方程也是有代数解法的，只是中学里一般不学罢了．但是，数学家已经证明，五次方程和更高次的方程，就找不到普遍适用的代数解法．这就是说，不会有用方程系数经有限次加减乘除和开方运算把方程的根表示出来的公式．这种“无公式解”的本性，是和五次以下的方程不同的．由于这个原因，在本书中，以后我们只把五次和高于五次的代数方程叫做高次方程．

高次方程虽然没有普遍适用的代数解法，但是却有一些非代数的或者说非公式的解法．这一节，先介绍高次方程的不动点迭代解法．

代数方程都可以表示成

$$f(x)=a_0x^n+a_1x^{n-1}+\cdots+a_{n-1}x+a_n=0,\quad a_0\neq 0,$$

这里 $f(x)$是一个 n 次多项式．如果能够把方程

$$f(x)=0$$

改写成

$$x=\varphi(x)$$

的形式，并且能够找到一个 x^* 使得

$$x^*=\varphi(x^*),$$

那么，x^*就是原代数方程的一个解．

把方程 $f(x)=0$ 改写成 $x=\varphi(x)$的形式，非常容易，也有许多方式．例如，可以写成

$$x=-a_0x^n-a_1x^{n-1}-\cdots-(a_{n-1}-1)x-a_n,$$

这时 $\varphi(x)=-a_0x^n-a_1x^{n-1}-\cdots-(a_{n-1}-1)x-a_n$，

也可以写成

$$x=-\frac{a_1x^{n-1}+\cdots+a_{n-1}x+a_n}{a_0x^{n-1}},$$

这时，右端的分式函数就是所说的 $\varphi(x)$. 因为 $x=\varphi(x)$ 是从 $f(x)=0$ 变形来的，所以新方程的解就是原方程的解.

x^* 是新方程的解，就是说 $x^*=\varphi(x^*)$. 请看函数 $\varphi(x)$. 一个函数，就表示一个对应，或者说表示一个变换. 函数 φ 是把 x 变成 $\varphi(x)$ 的对应. 现在 $x^*=\varphi(x^*)$，就是说函数 φ 把 x^* 变成 $\varphi(x^*)=x^*$ 自己，换一个说法，就是 x^* 经过 φ 这个变换没有动. 由于这个原因，使得 $x^*=\varphi(x^*)$ 的点 x^* 叫做函数 φ 的**不动点**，形如 $x=\varphi(x)$ 的方程，也就叫做**不动点方程**.

从上面可以看出，把代数方程改写成不动点方程是容易的，难的是怎样得到不动点 x^*. 为此，我们采用**迭代**办法：找一个点记作 x_0，代入函数 φ，得到 $\varphi(x_0)$，记作 x_1，再代入 φ，得到 $\varphi(x_1)$，记作 x_2，……如此一直做下去，可以得到一个序列

$$x_0, x_1, x_2, \cdots,$$

其迭代关系可以表示成

$$x_{n+1}=\varphi(x_n), \quad n=0,1,2,\cdots$$

有趣的是，这个迭代序列有时候可以帮助我们找到所要的不动点. 这就是**不动点迭代方法**.

先试一试.

考虑 5 次方程

$$x^5-17x+2=0,$$

首先把它变成一种不动点方程：

$$x=\frac{x^5+2}{17}\equiv\varphi(x),$$

这里的≡表示 $(x^5+2)/17$ 就是 $\varphi(x)$. 选 $x_0=0$ 进行迭代，得

$$x_1=2/17=0.117647,$$

$$x_2=(0.117647^5+2)/17=0.1176483,$$

$$x_3=(0.1176483^5+2)/17=0.1176483,$$

$x^* = 0.1176483$ 就是 φ 的一个不动点，所以是原 5 次方程的一个解.

熟悉这方面内容的读者可能先已看出，2 是原 5 次方程的一个解. 但是如果你不懂迭代法，或者虽然懂但是不去做，就无论如何看不出 0.1176483 这个解.

这个迭代过程要算 5 次方，好在一般计算器都有这个功能. 所以，每次迭代只是按几下键的事情，这是科技发展给我们带来的好处. 有些同学只喜欢推理，不喜欢计算，更不喜欢小数点，认为计算没有多少学问. 其实，计算和迭代都有深刻的学问. 许多重大的科学现象，都是在计算机帮助下发现的. 美国物理学家菲根鲍姆"玩"计算器入了迷，为开创混沌理论作出了重大贡献，就是生动的例子. 小小计算器空有许多功能而我们不去利用，岂不可惜？

刚才，我们选 $x_0=0$ 开始迭代，获得成功，这是不是巧合？是不是接受了什么暗示？提出怀疑是完全合理的，应当多做几次试验. 下面，分别从 $x_0=1$，$x_0=-1$，$x_0=2$，$x_0=-2$ 开始迭代，把 4 个迭代序列记录在一张表上：

n	x_n	x_n	x_n	x_n
0	2.0	1.0	−1.0	−2.0
1	2.0	0.1764705	0.0588235	−1.7647059
2		0.1176571	0.1176471	−0.8890822
3		0.1176483	0.1176483	0.0849686
4		0.1176483	0.1176483	0.1176473
5				0.1176483
6				0.1176483

到现在为止，5 个迭代序列都是成功的，一共找到 2 个解. 下面，再扩大范围试试，从 $x_0=3$ 和 $x_0=-3$ 开始迭代，数据如

下表：

n	x_n	x_n
0	3.0	−3.0
1	14.411765	−14.176471
2	36571.122	−33681.402
3	3.84806×10^{21}	-2.5497×10^{21}

不必再算下去就可以判断，这两个序列都是没有极限的．迭代公式是 $x_{n+1}=\varphi(x_n)=(x_n^5+2)/17$．当 x_n 已经很大时，x_n^5 就会非常大，最后除以 17，仍然保持很大．所以，从 $x_0=3$ 开始的迭代跑向 $+\infty$，从 $x_0=-3$ 开始的迭代则趋向于 $-\infty$，它们都不收敛．这时，我们也说这样的迭代序列发散．

这一节开头说过，不动点迭代方法一学就会．的确，现在你已经会了．但是我们没有说会做迭代就保准迭代成功．迭代是否成功，怎样使迭代成功，正是数值计算的复杂性要讨论和研究的问题．

为了进一步把上述迭代的情况研究清楚，可以画一张迭代收敛图帮助我们分析．

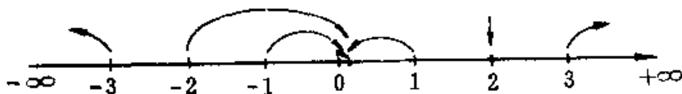


图 1.1

图 1.1 标明了前面所做的 7 个迭代过程的结果，1 个迭代驻守在 2 这个不动点，4 个迭代收敛到 0.1176483，另外 2 个迭代分别向 $+\infty$ 和 $-\infty$ 发散．这个图启发我们提出进一步的问题：

(1) 从 −3 开始的迭代发散．从 −2 开始的迭代收敛．−3

和－2之间，应当有一个分界点. 分界点在哪里？从分界点开始的迭代，究竟怎样进行？

（2）从点1开始的迭代收敛到0.1176483这个不动点，从点2开始的迭代驻守在点2这个不动点，它们之间也应当有一个分界点，也有同样的问题.

（3）点2和点3之间，也存在同样的问题.

为此，我们做3个迭代，数据如下：

n	x_n	x_n	x_n
0	－2.1	1.9	2.1
1	－2.2847653	1.5741759	2.5200594
2	－3.5446962	0.6862608	6.0963224
3	－32.801317	0.1266006	495.44315
4	－2233610.6	0.1176489	
5		0.1176483	
6		0.1176483	

看来，点2向右过去一点，迭代就发散到＋∞，点2向左一点，迭代就收敛到0.1176483；而从点－2.1开始的迭代，就发散到－∞.

更细致的试验，得出以下数据：

n	x_n	x_n	x_n	x_n
0	－2.0590	－2.0589	1.9999	2.0001
1	－2.0592245	－2.0586959	1.9995295	2.0004706
2	－2.0604117	－2.0576176	1.9977868	2.0022158
3	－2.0666974	－2.0519269	1.9896078	2.0104505
4	－2.1002199	－2.0220904	1.9516012	2.0496956
5	－2.2860233	－1.8709824	1.7830008	2.2457759
6	－3.5547898	－1.2310038	1.1776542	3.4779865
7	－33.272681	－0.048636	0.2508887	30.053423
8	－2398777.8	0.117647	0.1177055	1442184.5
9	-4.6719×10^{30}	0.1176483	0.1176483	
10		0.1176483	0.1176483	

从以上试验，可以得出两个结论．一，点 2 是个孤立的很不稳定的不动点，迭代出发点 x_0 与点 2 差一点点，迭代结果就完全不同．二，问题 (2) 的分界点，在 -2.0590 和 -2.0589 之间．现在我们用另一种不但一学就会而且必定成功的迭代法把这个分界点定下来，这就是**对分区间迭代法**．现将这种方法叙述如下：

我们要解的是方程 $f(x)=0$. 如果我们已经知道两点 $a<b$ 使得 $f(a)\cdot f(b)<0$，即 $f(a)$ 和 $f(b)$ 符号相反，那么在 (a,b) 当中取中点 c，计算 $f(c)$. 倘若 $f(c)=0$，解已求到，不必再迭代下去．否则，如果 $f(a)\cdot f(c)<0$，知 $f(c)$ 和 $f(b)$ 同号，就扔掉原来的 b，把 c 作为新的 b，仍如上法迭代；如果 $f(c)\cdot f(b)<0$，知 $f(c)$ 和 $f(a)$ 同号，就扔掉原来的 a，把 c 作为新的 a，仍如上法迭代．这就是同符号顶替的原则．这样，每迭代一次，(a,b) 区间的长度缩小一半，而在区间两头，函数 $f(x)$ 的符号总是相反．于是，根据 $f(x)$ 的连续性（这点可暂不深究），这些每次缩短一半的区间最后套住一个点，这个点一定是 $f(x)=0$ 的解．

现在就来试试．前已知道，-2.0590 和 -2.0589 之间应当有一个分界点，我们就拿 $a=-2.0590$ 和 $b=-2.0589$ 开始．$f(a)=-3.817\times10^{-3}<0$，$f(b)=3.469\times10^{-3}>0$，正好符合 $f(a)\cdot f(b)<0$. 取 (a,b) 的中点 $c=-2.05895$，算 $f(c)$，这时不必记录具体结果，只要知道正负就可以．算得 $f(c)<0$，与 $f(a)$ 同号，所以按照同符号顶替的原则，取 c 作新的 a，下次迭代就取 $(a,b)=(-2.05895,-2.05890)$.

如果拘泥于中点，下次该取 $c=-2.058925$. 但对分区间迭代有个好处，c 取偏一点关系不大，只要不跑出 (a,b) 就行．为了不一下子增加数字长度，我们取 $c=-2.05893$，算得 $f(c)>0$. 再取 $c=-2.05894$，也得 $f(c)>0$. 接下去：

$$c=-2.058945, f(c)>0,$$
$$c=-2.058947, f(c)>0,$$
$$c=-2.058948, f(c)<0,$$
$$c=-2.0589475, f(c)>0,$$
$$c=-2.0589477, f(c)<0,$$
$$c=-2.0589476, f(c)>0.$$

限于计算器的能力，再精确下去已经不可能. 我们就取 $c=-2.0589476$作为 $f(x)=0$ 的一个近似解. 这时，$f(c)=1.2\times 10^{-6}$，这个解已经相当精确.

至此，我们找到了 $f(x)=0$ 的三个解，也就是 $x_{n+1}=\varphi(x_n)$ 迭代的三个不动点，即 2，0.1176483 和 -2.0589476. 经过这样一番研究，我们对 $x_{n+1}=\varphi(x_n)$ 迭代的收敛情况有更准确的了解. 这些情况，都总结在图 1.2 上.

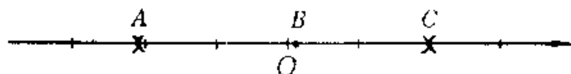


图 1.2

方程 $f(x)=x^5-17x+2=0$ 共有 3 个不同的实数解，它们是 $A\doteq-2.0589476$，$B=0.1176483$ 和 $C=2$. 有的读者可以看出 C 这个解. 但是如果不做迭代法，再聪明的人也不会知道 A 和 B 这两个解.

从不动点迭代 $x_{n+1}=\varphi(x_n)=(x_n^5+2)/17$ 的角度来看，A 和 C 都是孤立的不稳定的不动点，在 A 或 C 附近开始进行不动点迭代，出发点差一点点，迭代结果就会面目全非. 如果以迭代的结局来瓜分数轴，那么 $(-\infty, A)$ 是 $-\infty$ 的地盘，(A, C) 是 B 的地盘，$(C, +\infty)$ 是 $+\infty$ 的地盘，而 A 的地盘只有它自己一个点 $\{A\}$，C 的地盘也只有它自己一个点 $\{C\}$.

请记住这个例子，后面，它还会给我们提供新的启示.

§2 收敛性和复杂性——算法优劣判别的两个层次

数学讨论最终还是要解决实际问题. 如果面对的是方程求解的问题，那么首先就要回答方程有没有解这个所谓解的存在性的问题. 方程有多少个解，解在什么地方等等，也都属于这类问题.

存在性讨论有两种基本方式. 一种是构造性的证明方法，那就是具体设计一种方法把解找出来，从而说明解是存在的. 找到了的东西当然是存在的东西，这就是构造性方式的哲学. 另一种是非构造性的证明方法，其代表就是反证法：假定没有解，然后通过推理，引出与已知事实的矛盾.

反证法在逻辑上常常是漂亮的，但是带给人们的信息较少. 例如，一所2千人的中学，总有1千人的身高在1.5米和2米之间，由此容易推出至少有两个人的身高相差不到半毫米. 这里，反证法所用的，也算是抽屉原理. 如果要你通过实测找出身高相差不足半毫米的两个人来，就不那么容易了. 又比如，面对3只雏鸡，看也不看你就可以作出“其中必有两只雏鸡的性别相同”这样一个存在性的判断，因为不然的话，就和鸡只有雌雄两个性别的已知事实矛盾. 这个判断论证过程，就是非构造性的. 倘若你是一个识鸡的行家，确实辨认出其中有两只是雄雏，并由此得到“有两只雏鸡的性别相同”的判断，这个判断论证过程就是构造性的. 两相比较，构造性的判断方式具体指出了两只雄雏，所以提供的信息比较多，而前面说的非构造性的判断，在我们这个雏鸡性别问题里，一点也没有提供出有

实用价值的信息.

但是在数学发展的漫长进程之中，非构造性的讨论方法作用很大，功不可没. 我们知道，社会要求和内部动力是数学发展的两大激励. 非构造性的讨论方式，常常就是数学大系统的内部动力的体现. 另外，除了未有构造性方法时自然欢迎非构造性方法以外，我们还要注意，人类的认识都是相互联系的，非构造性方法得到的结论，常常可以给研究构造性方法指引方向. 最典型的例子，就是数学家已经用非构造性的方法证明了，不存在用圆规直尺三等分任意角的通用方法，不存在高次方程的代数解法，这就使后人可以避免在寻求三等分角和高次方程代数解法方面空耗精力（论证某种东西不存在，和论证某种东西存在一样，都属于存在性问题）. 相反，如果对于一个问题，数学家已经用非构造的方式证明了解是一定存在的，这就指引后人更明确地去寻找把解具体找出来的构造性方法. 对分区间迭代法就是这样想出来的.

应用部门对数学的要求，主要不在于解是存在的这样一种结论，要紧的是把解具体找出来. 从这个角度看，构造性的方法虽然做起来有时比较辛苦，却不但肯定了"存在"的事实，还告诉人们怎样把这个"存在"找出来. 如果说有些构造性方法因为计算比较繁复，过去还很难应用的话，现在由于电子计算机的发展和普及，"繁复"的弱点大半已经不成问题，构造性方法正好可以借助计算机避短扬长，向人们提供满意的答案. 所以，随着科学的发展，各方面对数学的要求，越来越倾向于构造性的解决办法. 面对方程求解的问题，构造性的工作，就是要寻找把解算出来的方法，或者说寻找求解的算法.

既然我们强调构造性工作的实际应用价值，那么，面对一种算法，第一要问它是否成功，第二要问它效率如何. 不成功

的算法不能把解求出来，当然没有用．成功但是效率很差的算法，也没有多少价值．如果有人告诉你一种算法，并且在数学上证明了按这种算法一定可以找到问题的解，但是求解要在最新的电子计算机上花费多少万年的时间，你对这样一种实际上无法实现的构造性方法会有什么感想？恐怕是啼笑皆非吧？后面，会谈到一些这样的算法．

算法是否成功，是**收敛性**问题．收敛的就成功，不收敛的即发散的就不成功．效率如何，是算法的**复杂性**问题．复杂性问题是本书的主题，我们会在以后的章节逐步展开．

本书后面谈得比较多的方法，是代数方程的牛顿算法和库恩算法，它们都属于迭代方法．事实上，当今人们用电子计算机进行科学计算所采用的方法，大部分都是某种类型的迭代方法．现在，我们用几个很简单的不动点迭代，说明收敛性和复杂性的意义．

考虑代数方程 $f(x)=x^2-a=0$，这里 $a>0$．这个方程太简单了．但是，着眼于展示收敛性和复杂性的意义，我们为什么非要复杂的方程不可呢？找复杂的方程来演示，会本末倒置，花费许多精力到技术细节上，妨碍我们对问题实质的认识，所以，这里宁愿用简单方程．

我们采用 4 种方案，把 $f(x)=0$ 变成不动点方程：

(1) $x=x+(x^2-a)$；

(2) $x=a/x$；

(3) $x=(x+a/x)/2$；

(4) $x=x+(x^2-a)/4$．

把不动点方程右端的表达式看作 $\varphi(x)$，就可以将迭代公式 $x_{n+1}=\varphi(x_n)$ 具体写下来：

(1) $x_{n+1}=x_n+(x_n^2-a)$；

(2) $x_{n+1}=a/x_n$;

(3) $x_{n+1}=(x_n+a/x_n)/2$;

(4) $x_{n+1}=x_n+(x_n^2-a)/4$.

4 个迭代都从 $x_0=2$ 开始，迭代情况如下：

n	(1)	(2)	(3)	(4)
0	2.0	2.0	2.0	2.0
1	3.0	1.5	1.75	1.75
2	9.0	2.0	1.732143	1.734375
3	87.0	1.5	1.732051	1.732361
4			1.732051	1.732092
5				1.732056
6				1.732051
7				1.732051

迭代(1)发散向 $+\infty$，不收敛，迭代(2)振荡，也不收敛，迭代(3)和迭代(4)都收敛到方程的解. 虽然(3)和(4)都收敛，但是很明显，迭代(3)的效率比迭代(4)好，换句话说，迭代(3)收敛得比迭代(4)快.

下一节，我们会论述收敛快慢的标准. 现在，借上面的例子说一件有趣的事情.

读者是否注意到，迭代(1)和迭代(4)都可以表示成统一的形式，那就是

$$x=x+c(x^2-a).$$

当取 $c=1$ 时就得(1)，迭代不收敛；当取 $c=1/4$ 时就得(4)，迭代就收敛了. 这个 c 的选择，很有讲究. 选得好，就收敛，甚至收敛很快；选得不好，就算得很慢，甚至根本不收敛. 读者有计算器，不妨自己选 $c=1/2$ 或 $c=1/8$ 等等试一试.

如何选 c 的原则，本书就不讲了，那是计算数学的一个容易

入门的并且很有价值的课题. 更推而广之，怎样把 $f(x)=0$ 变成 $x=\varphi(x)$，里面的项怎么拆好，有些可以自由选取的系数(例如上面的 c) 怎么选好，都是科学研究的对象. 运用之妙，存乎一心，读者都是科学的有心人.

§3 可怕的指数增长——古印度数学故事

效率之高低，指的是收敛之快慢，这是没有疑问的. 倘若我们知道，用算法 A 求问题 Y 的解要 10 秒钟，用算法 B 求问题 Z 的解要 3 分钟，你能下结论说算法 A 的效率比算法 B 高吗?

不能这样下结论. 首先，你用每秒可运算亿次的银河机算了 10 秒钟，我用可编程序的计算器算了 3 分钟，机器功能差得太远，决不能说算法 A 算得比算法 B 快.

即使机器完全一样，也不能贸然下结论说算法 A 比算法 B 效率高. 假如 Y 是 $x-1.7=0$ 这样一个方程的求解问题，而 Z 是 $x^5-17x+2=0$ 这样一个方程的求解问题，那么 Z 比 Y 难得多. 算法 A 解一个容易的问题需要 10 秒钟，算法 B 解难得多的问题需要 3 分钟，并不能说明算法 A 比算法 B 效率高.

机器的问题后面再讨论，现在只说说问题之难易. 难度这个概念，是主客观两方面因素的共同反映. 一元二次方程对初一的学生很难，对初三的学生就相当容易. 为了剔除主观能力因素的影响，我们舍弃难度这个概念，而采用问题的**规模**这个概念. 我们不忙于下定义，只从具体问题说起. 面对代数方程求根这样的数值计算问题，方程的次数，就是问题规模的最主要的指标. 除了特殊方程的例外情况以外，解 9 次方程当然比

解 8 次方程工作量要大，这点大家都会同意.

同一种算法，解小规模的问题花时间少，解大规模的问题花时间多，这是很自然的事. 问题是，随着问题规模的增加，所需要的计算时间怎样增加？以代数方程求解的问题来说，如果方程的次数是 n，求解所需要的时间，我们把它暂记作 $T(n)$. $T(n)$ 究竟是 n 的什么函数呢？即使没有明确的函数关系，也要尽可能把它们的关系反映出来.

对于复杂性问题来说，最要命的关系是形如 $T(n)=a2^n$ 的**指数关系**. 指数关系为什么这么要紧？我们先看一个古印度的数学故事.

传说宰相达依尔因为发明了国际象棋很得印度舍尔王的欢心. 舍尔王打算重赏达依尔，问他有什么要求，达依尔跪在国王面前说："陛下，请你在这张棋盘的第 1 个小格内，赏给我 1 粒小麦，在第 2 个格子内给 2 粒，第 3 个格子里给 4 粒，第 4 个格子里给 8 粒，照这样下去，每一小格内的麦粒都比前一小格内多一倍. 陛下啊，当这棋盘上的 64 个小格都摆完麦粒的时候，你就把这些麦粒都赐给你的仆人吧！"

大家知道，国际象棋共有 64 个黑白格子. 国王听了达依尔的要求，觉得他的要求实在很低，几粒麦子有什么了不起？说着，国王就令人把一袋小麦背到宝座前.

计数麦粒的工作开始了. 第 1 格放 1 粒，第 2 格放 2 粒，第 3 格放 4 粒，……. 还没到第 20 格，袋子已经空了. 一袋又一袋小麦被扛到国王面前来. 但是，麦粒数目一格接一格增长得那么快，人们马上意识到，即便把全印度的小麦也搬来，国王也兑现不了他对达依尔的赏赐. 国王感到丢脸，砍了达依尔的脑袋.

让我们来算一下，达依尔所要的麦子，究竟有多少？其实，

麦粒数目正是

$$1+2+2^2+2^3+\cdots+2^{63},$$

这是公比为 2 的一个等比数列的前 64 项的和，它等于

$$\frac{2^{64}-1}{2-1}=2^{64}-1$$
$$=18,446,744,073,709,551,615.$$

查一下植物学的书，麦子不是说每粒多重，而是说每千粒多重。一般小麦的千粒重大约是 40 克，所以，达依尔要求的小麦，重约

$$1.8\times10^{19}\times40/1000 \text{ 克}$$
$$=1.8\times40\times10^{10}\text{吨}$$
$$=720,000,000,000\text{ 吨}.$$

全世界在两千年里生产的小麦加起来，也不到这个数。可见，这位聪明的宰相看起来很微不足道的要求，竟有这么大的份量！

从 1 增加到 2，从 2 增加到 4，都很不显眼。但是接着做下去，没几下，就变成了天文数字，这就是指数增长的可怕之处。算法效率高低的最要紧的标志，就是解题时间 $T(n)$ 随着问题规模 n 的增长，绝对不可以是指数式的增长关系！

古代印度的数学家，可能是世界上最早领会到指数增长之厉害的人，棋盘放麦子的故事就是一个例子。下面再讲一个梵天宝塔的传说，它明确地把一种算法与计算时间的指数增长展现在我们面前。

古代印度人把佛教圣地贝拿勒斯看作是世界的中心。传说在贝拿勒斯的圣庙里，有一块黄铜板，上面竖着 3 根宝石针。这些宝石针，径不及小指，长仅可半臂。大梵天王（印度教的一位主神）在创造世界的时候，在其中一根针上，放置了 64 片中心有插孔的金片。这些金片大小都不一样，大的在下面，小的

在上面，从下到上叠成塔形，这就是所谓梵天宝塔.

大梵天王为梵塔立下了至尊不渝的法则：这些圆形金片可以从一根针移到另一根针，每次只能移动一片，并且要求不管在任何时刻，也不管在哪根针上，小金片永远在大金片上面，绝不允许颠倒.

大梵天王预言，当叠成塔形的 64 片金片都从他创造世界时所放置的那根针上移到另一根针上去、并且也是大的在下小的在上叠成塔形的时候，世界的末日就要来临，一声霹雳将把梵塔、庙宇和众生都消灭干净.

如果不是数字 64 在暗示的话，读者可能会想：这个传说未免太不高明了，不就是 64 个金片吗？顶多几个钟头就可以实现宝塔挪位，到那个时候，预言者岂不是要自己掌嘴？

我和你一样，不相信什么世界末日．不过，按照梵天的规则把宝塔从一根针上搬到另一根针上，可不是容易的事．诚然，传说毕竟是传说，但我们不妨把梵天宝塔作为一种数学游戏，自己动手试试．“实践出真知”，这可是学问的至理.

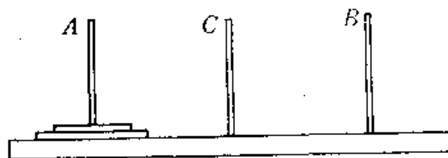


图 1.3

按照梵天的规则，把一个 n 层的宝塔从 A 柱移到 B 柱，至少要移动多少次呢？我们把这个移动次数记作 $S(n)$．$n=64$ 是比较多的，我们先从 $n=1$，2，3 做起.

$n=1$ 时，把金片直接从 A 柱移到 B 柱就行了，所以 $S(1)=1$．$n=2$ 时，可以借助 C 柱．先把小片移到 C 柱暂住，再把

大片直送 B 柱，最后把小片也移到 B 柱上压在大片上面．3 步完成，所以 $S(2)=3$．$n=3$ 时，我们这样来分解任务：先把上面的 2 层塔移到 C 柱，再把最底片移到 B 柱，最后把那个 2 层塔移到 B 柱，这样，3 层塔的挪位完成．前已讨论过 2 层塔，知道前后将 2 层塔从 A 移 C 和从 C 移 B 都各需 3 次，当中一步将底片从 A 移 B 只需 1 次，所以移动次数是 $3+1+3=7$ 次，知 $S(3)=7$．

同样道理，$n=4$ 时，$S(4)=7+1+7=15$．这启发我们猜想 $S(n)=2^n-1$．下面就来完成这个猜想的归纳证明．

设公式对 $n=k$ 已成立，现在要移 $k+1$ 层宝塔，任务三段分解：先将上面 k 层由 A 柱移到 C 柱暂居，需要 2^k-1 步；再将最底下的大片由 A 柱直移 B 柱，需 1 步；最后将上面的 k 层宝塔从 C 柱送到 B 柱，需要 2^k-1 步．由此可得：

$$S(k+1)=2^k-1+1+2^k-1=2^{k+1}-1.$$

归纳证明完成．

至此我们知道，把一个 n 层梵塔按照梵天的规则从一根柱上搬到另一根柱上，至少要移动金片 $S(n)=2^n-1$ 次．

假如你手脚非常麻利，一秒钟可以移动一次金片，那么：

$n=1$ 时，1 秒钟你就完成任务，

$n=2$ 时，3 秒钟你就完成任务，

$n=3$ 时，7 秒钟就够了，

$n=4$ 时，需要 15 秒钟，

$n=5$ 时，31 秒，

$n=6$ 时，$(2^6-1)/60=1.05$ 分钟，

$n=7$ 时，$(2^7-1)/60=2.12$ 分钟，

$n=8$ 时，$(2^8-1)/60=4.25$ 分钟，

……

看起来都没有什么了不起，可是，当 $n=64$，变成真正的梵天宝塔时，尽管你训练有素，手脚麻利，也需要

$$S(64)=2^{64}-1$$
$$=18,446,744,073,709,511,615$$

秒的时间才能完成移塔的任务．我们知道，平均一年约 365.24 天，每天 24 小时，每小时合 $60\times60=3600$ 秒，所以一年大约 31，557，000 秒．由此可以算出，你需要大约

$$(2^{64}-1)\div31557000$$
$$\approx584,600,000,000$$

年的时间才能完成你的作业．

你面临的是宝塔移位的问题，问题的规模就是宝塔的层数 n．问题的规模只从 $n=1$ 增加到 $n=64$，你为了解决这个问题所需要的时间却从 1 秒钟增加到将近 6,000 亿年！

这就是指数增长的厉害．

按照近代天体物理学的一种理论，太阳系是在大约 30 亿年前由处于浑沌状态的物质逐渐演变而成的，太阳的核子燃料还能使用 100～150 亿年．如果这种理论是正确的，那么粗略一算，就知道太阳系的寿命不会超过 200 亿年．这些数据不应当成为“世界末日”的证明．远在太阳系的寿命结束以前，人类文明该早已找到新的寄托，新的载体，新的可以更加大显身手的广阔天地．值得我们惊叹的是，古印度的先贤们对指数增长竟有如此深刻的了解．64 是一个小小的很平凡的数目字，可是通过指数关系，64 层梵天宝塔所赋予的期限，原来却如此之长．这个期限比太阳系的寿命还长得多，谁还能企求更多呢？

§4 寻求多项式时间算法

上一节我们谈的是一个问题的一种解法，问题是“梵塔移柱”，解法我们把它叫做“借柱归纳”．这是因为把塔从 A 柱移到 B 柱的过程中，要借助 C 柱让一些金片暂居，移柱的方法是归纳地说明的．问题的规模是金片的数目 n．解法的工作量，以移片次数表示，是 $S(n)=2^n-1$．也许你对“借柱归纳”解法不满意，因为 $n=64$ 时就要花 6000 亿年才能完成任务，但是可以证明，这已经是最经济最聪明的解法了．工作量大，是问题本身的性质所规定的，在这样的问题面前，不会有更好的解法．

现在我们换一个角度看看．假如面对同一个问题，几个人使用几种不同的解法，我们要评判哪一种解法最节省，最经济．

仍设问题的规模为 n．假如有 6 种不同的解法，它们要解决这个规模 n 的问题所需要的工作量分别是

$$S_1(n)=n,\qquad S_2(n)=n^2,$$
$$S_3(n)=n^3,\qquad S_4(n)=n^5,$$
$$S_5(n)=2^n,\qquad S_6(n)=3^n.$$

这些 $S(n)$ 代表了解题的工作量，反映解法的效率如何．因为工作量的大小，工作效率的高低，总是可以用工作时间来表示的，所以上述 $S(n)$ 函数，通常就叫做解法或算法的**时间复杂性函数**．

上述 6 个时间复杂性函数中，S_1 是一次函数或线性函数，S_2 是 2 次函数，S_3 是 3 次函数，S_4 是 5 次函数．它们统称为**多项式**时间复杂性函数，这是因为它们都可以成为以 n 为变元的某个多项式的一部分．与此相对照，S_5 和 S_6 这样的时间复杂性

函数，就叫做**指数式**时间复杂性函数．由于当 n 很大时，2^n-1 和 2^n 之比几乎就是 1，所以“梵塔移柱”的“借柱归纳”解法的时间复杂性函数，也是指数式时间复杂性函数．

现在比较一下这 6 种方法的解题速度．假定都用 1 微秒(0.000001 秒)能够完成 1 次运算(例如移动一片金片这样的动作)的高速电子计算机，那么问题规模 n 和解题时间之间的关系，可以列成下面的表格：

时间复杂性函数	问题规模 n				
	5	10	20	40	60
n	0.000005 秒	0.00001 秒	0.00002 秒	0.00004 秒	0.00006 秒
n^2	0.000025 秒	0.0001 秒	0.0004 秒	0.0016 秒	0.0036 秒
n^3	0.000125 秒	0.001 秒	0.008 秒	0.064 秒	0.216 秒
n^5	0.003125 秒	0.1 秒	3.2 秒	1.7 分	13.0 分
2^n	0.000032 秒	0.001 秒	1.0 秒	12.7 天	366 世纪
3^n	0.000243 秒	0.059 秒	58 分	3855 世纪	1.3×10^{13} 世纪

从表上可以看出，对于多项式时间复杂性函数，工作量随问题规模 n 增长而增长的速度都比较温和，但是对于指数式时间复杂性函数，这种增长到后来就非常激烈．我们特别比较一下 $S_4(n)=n^5$ 和 $S_5(n)=2^n$ 这两行．当 n 比较小时，2^n 甚至比 n^5 还节省，但当 n 较大时，2^n 方法所需的解题时间就比 n^5 方法所需的时间大得多．例如 $n=40$ 时，n^5 方法只需要 1．7 分钟，而 2^n 方法需要 12.7 天；当 $n=60$ 时，n^5 方法只需要 13 分钟，而

2^n 方法却需要 366 个世纪！

表列的 $n=60$，还不如上一节讨论的 $n=64$. 随着社会经济的发展和科学技术的发展，应用问题的规模数达到几百几千是常有的事. 例如，投入产出的经济分析，就经常要对付几百个经济变量，这个"几百"，就是投入产出分析问题的规模. 本书后面会谈到线性规划问题. 大型线性规划问题要对付上千个方程（"等式"）和上千个约束条件（"不等式"），这里的"几千"，就是线性规划问题的规模.

读者也许会想：虽然问题规模越来越大，但是计算机的运算速度也越来越大，所以没有什么可担心的. 其实不然.

我们还是以上述 6 种方法作为例子，试作分析. 假定这 6 种方法用现有的计算机在一小时内可以解决的问题的最大规模都是 $n=1000$. 这样做，是为了把 6 种方法都放在同一条起跑线上，便于做公平的比较. 下面的表告诉我们，如果发明了速度等于现有机器 100 倍或 1000 倍的计算机，那末在一小时内可解的问题的规模如何变化？①

一小时内可解的问题的最大规模			
时间复杂性函数	用现有的计算机	速度 100 倍的计算机	速度 1000 倍的计算机
n	1000	100000	1000000
n^2	1000	10000	31600
n^3	1000	4640	10000
n^5	1000	2500	3980
2^n	1000	1007	1010
3^n	1000	1004	1006

① 读者可自行推算出表中最后两列的全部约数，这是学习中学数学指数关系的一个训练.

从表上可以看到，当发明了新的计算机，运算速度提高到原来的 1000 倍时，对于 n^5 方法，一小时内可解的问题的规模从 1000 增加到 3980，差不多是原来的 4 倍，而对于 2^n 算法，可解问题的规模从 1000 增加到 1010，只增加百分之一.

由此可见，面对随着社会经济发展和科学技术发展而规模日益扩大的问题，指数式时间复杂性函数的计算机解法，是力不从心，无能为力的. 相反，多项式时间复杂性函数的计算机解法，却能胜任自如.

任何问题的计算机解法，都通称为算法. 具有多项式时间复杂性函数的算法，简称为**多项式时间算法**；具有指数式时间复杂性函数的，简称为**指数时间算法**，但是要注意指数时间算法也包括某些原来不看作指数函数但肯定不是多项式的情形，例如 $n^{\log_2 n}$. 所以，复杂性理论中所说的指数时间算法，可以理解为非多项式时间算法的算法.

基于上述讨论分析，我们就可以明白，为什么在计算机科学和计算数学中，都把多项式时间算法看作是“好的”算法. 科学研究的一个重要内容，就是判别和论证现有的各种算法是不是多项式时间算法，或者寻找和设计新的多项式时间算法，这就是本节小标题的份量.

既然多项式时间算法是好的算法，那末，是否凡指数时间算法都是不好的算法呢?

这个问题不那么简单. 回忆第一节讲的不动点迭代

$$x_{n+1}=(x_n^5+2)/17,$$

这是解方程 $x^5-17x+2=0$ 的一种算法. 读者已经体会到，这个算法相当好，它使你借助一个袖珍计算器，在短短几分钟内就求出了方程的全部不同的实根，其中 $x=-2.0589476$ 和 $x=0.1176483$ 是你怎么也猜不着的. 但是这个迭代算法是多项式

时间算法吗？

不是.

多项式时间算法的**定义**是：解决规模为 n 的问题所需要的时间不超过 cn^k，这里 c 是一个正常数而 k 是一个固定的非负整数. 我们记得，上述迭代如果从 $x=3$ 开始，就根本不收敛，不解决问题. 这就是说，不论你迭代多少时间，问题总不能解决. 所以，按 cn^k 确定时限的 c 和 k 不存在，这就说明它不是多项式时间算法.

但是如果从比较好的地方开始迭代，它又算得很好，十分令人满意，所以，面对一个非多项式时间算法，我们并不立即判定它是不好的算法而完全抛弃，而是要研究它什么时候不好，为什么不好，更要研究它什么时候会表现出好的行为，可以解决我们面对的问题.

本来，只有当一种算法是收敛的时候，才可以说它是多项式时间算法或指数时间算法. 上述不动点迭代从很多地方开始迭代都不收敛，我们尚且不能笼统说它不好，那末如果是收敛的算法但收敛得比较慢（指数时间），更不能马上说它不好. 这是一个深一步的问题，我们在第三章再讲. 在目前这一章，首先要知道的是什么是多项式时间算法和它为什么那么重要.

§5 温故而创新的代数基本定理

第二章和第三章，我们将分别叙述多项式求根的库恩算法和牛顿算法，着重讨论它们的复杂性问题. 具体地说，就是要研究这两种算法是不是多项式时间算法.

两种算法自然都是"多项式求根算法"，我们却要研究它们

是不是“多项式时间算法”．请注意，这在概念上是很不相同的两个名称．多项式求根算法，是就算法的目标而言的，表明这些算法用于多项式求根．是否多项式时间算法，是就算法的性状、效率、复杂性而言的，判断它们是否上一节所说的“好的”算法．

科学计算的问题很多，为什么偏偏挑出多项式求根问题的算法来研究呢？这至少有两方面的原因．第一，多项式是大家很熟悉的数学对象，计算复杂性讨论是一种新的科学讨论，在一个比较熟悉的问题上开始一种崭新的讨论，有助于我们把注意力集中在新的因素上，迅速抓住问题的要害．第二，许多算法都不只是用于对付一种问题，而是可以对付一大类问题．拿牛顿算法来说，不仅用于多项式求根，即解一元高次代数方程，而且可以用于求解多元高次代数方程组，甚至用于求解多元超越方程组这样一些困难得多的所谓高度非线性问题．库恩算法也是这样．就人类的认识过程来说，了解这些算法在对付比较简单的问题时行为和效率如何，是了解它们在对付比较困难的问题时的行为和效率的先导．所以，本书重点论述的，就是两种算法在对付多项式求根问题时的效率如何，或者说复杂性程度．

在此，有必要把多项式和多项式的根讲个明白．当 n 是正整数的时候，形如

$$f(z)=c_nz^n+c_{n-1}z^{n-1}+\cdots+c_1z+c_0$$

的函数，称为 n 阶多项式．需要着重指出的是，这里 z 是复变量，c_0，c_1，$\cdots$，c_{n-1}，c_n 都是复常数，并且 $c_n\neq 0$．我们将在复数域或复数平面上展开讨论，这是和中学数学的侧重点不同的地方．好在中学阶段读者已学过复数及其运算，所以强调在复平面上讨论问题并不会引起原则性的困难．

如果一个复数 ξ 使得用 $z=\xi$ 代入上述多项式使多项式的值变为零，即使得

$$f(\xi)=c_n\xi^n+c_{n-1}\xi^{n-1}+\cdots+c_1\xi+c_0=0,$$

就说复数 ξ 是多项式 $f(z)$ 的一个根. 例如我们知道，$\xi=2$ 就是 5 阶多项式 $f(z)=z^5-17z+2$ 的一个根. 另外，$\xi=-2.0589476$ 和 $\xi=0.1176483$ 也是多项式 $f(z)=z^5-17z+2$ 的根.

多项式有没有根，有多少个根，在近两百年前就已经清楚了. 我们讨论的是复系数多项式. 由于实数也属于复数，所以复系数多项式也包括了实系数多项式. 如果只限于实数范围，$f(z)=z^2+1$ 这样的多项式就没有根，但是放到复数范围里，每个多项式必有一个根. 论述这一事实的定理，就是著名的**代数基本定理**.

代数基本定理有两种不同的说法，不妨称为 A 形式和 B 形式. A 形式代数基本定理说，任一 n 阶（复系数）多项式必有一个（复）根；B 形式代数基本定理说，任一 n 阶（复系数）多项式正好有 n 个（复）根. 看起来，B 形式定理比 A 形式定理强，因为有 n 个根当然包括了有一个根. 但是实际上 A 形式和 B 形式是等价的，因为利用带余除法，只要肯定多项式必有根（A 形式），就可推出 n 阶多项式恰有 n 个根（B 形式）. 具体论证如下：

设 $f(z)$ 是 n 阶多项式，按照 A 形式的代数基本定理，$f(z)$ 必有一个根，把它记作 ξ_1；现在用 $z-\xi_1$ 除 n 阶多项式，得到的是一个 $n-1$ 阶多项式，根据 A 形式的定理，这个 $n-1$ 阶多项式也必有一个根，把它记作 ξ_2；再用 $z-\xi_2$ 除 $n-1$ 阶多项式，得到一个 $n-2$ 阶多项式，……这样一次一次做下去，除 $z-\xi_1$，$z-\xi_2$，…，$z-\xi_{n-1}$ 这样 $n-1$ 次以后，得到一个 1 阶多项式，把

它记作 $z-\xi_n$，这时我们就知道，原多项式可以写成

$$f(z)=(z-\xi_1)(z-\xi_2)\cdots(z-\xi_{n-1})(z-\xi_n),$$

而 ξ_1，ξ_2，…，ξ_{n-1}，ξ_n 就正好是它的 n 个根．这就从 A 形式的代数基本定理推出了 B 形式的代数基本定理．

我们以后经常用的是 B 形式的定理．

细心的读者可能会问，如果 n 阶多项式的 n 个根 ξ_1，…，ξ_n 有相同的怎么办？这个问题提得好，从这个问题可以引伸出重根的概念．

假定 n 个根中把相同的放在一起，结果只有 k 个不同的根，那末当然 $k\leqslant n$．把这 k 个根记作 η_1，…，η_k，并设 ξ_1，…，ξ_n 中有 n_1 个和 η_1 一样，有 n_2 个和 η_2 一样，…，有 n_k 个和 η_k 一样，那末就可以将

$$f(z)=(z-\xi_1)(z-\xi_2)\cdots(z-\xi_n)$$

改写成

$$f(z)=(z-\eta_1)^{n_1}(z-\eta_2)^{n_2}\cdots(z-\eta_k)^{n_k},$$

其中 n_1，n_2，…，n_k 都是正整数，可以是 1，但必须满足 $n_1+n_2+\cdots+n_k=n$ 的关系．

在这样整理以后，我们说 η_1 是多项式的 n_1 **重根**，n_1 是根 η_1 的**重数**．η_2 的 n_2，…，η_k 的 n_k 都有同样的意义．和重根相对照的概念是单根．在上面的分解式中，如果某个 $n_i=1$，就说 η_i 是多项式 $f(z)$ 的单根．从概念上说，重根既与单根相对，又包含单根．事实上，单根就是 1 重根．

在以后的讨论中我们会知道，重根常常会给我们带来麻烦．

这一节我们介绍了代数基本定理，但是没有给出证明．数学史上有一些经典的论题，它们虽然已经获得解决，却还是一再唤起人们的热情，去从事温故而创新的工作．关于多项式有没有根和有多少个根的代数基本定理，就是这样一个论题．大

家知道，德国数学家高斯从1799年到1850年前后跨越半个世纪的时间里，曾经提出代数基本定理的四种证明．更早，牛顿、麦克劳林、达朗贝尔、欧拉和拉格朗日，都从事过这一工作．但是，他们提供的证明，基本上是非构造性的证明，主要思路就是如果没有根，会引出什么样的矛盾．

科学和社会的发展，越来越倾向于构造性的方法．我们在第二章将讨论的库恩算法，就是具体告诉你怎样同时把 n 阶多项式的 n 个根都找出来的方法．库恩算法，就是代数基本定理的一种构造性的证明．

第二章　库恩算法及其计算复杂性

本章叙述库恩的多项式求根算法，以及这种算法的计算复杂性，即它的效率如何，是不是人们希望的多项式时间算法.

库恩（H. W. Kuhn）是美国普林斯顿大学数学系和经济学系的教授，是拓扑学、应用数学和数理经济学的专家. 他在 1977 年发表一篇论文，叙述和论证一种新的多项式求根算法，这就是现在通称的库恩算法.

库恩算法的好处是不用高等数学，可以向中学数学爱好者讲清楚. 有人可能认为不用高深数学的算法就不是高级的方法，其实不然. 对付同样的问题，你用很深奥很艰难的方法才能解决它，而我用较浅白较容易的方法就把它解决了，谁应该受到更多称赞？所以，在数学研究中，如果可能的话，人们总是使用不牵涉高深理论的所谓“初等方法”. 初等方法，才是许多人心目中的高级方法.

§1 库恩算法的描述

为了求多项式

$$f(z)=c_nz^n+c_{n-1}z^{n-1}+\cdots+c_1z+c_0,\quad c_n\neq0$$

的根，我们可以首先用 $c_n\neq0$ 除整个多项式. 设得到的多项式是

$$g(z)=a_nz^n+a_{n-1}z^{n-1}+\cdots+a_1z+a_0,$$

那末，二者之间的关系是

$$a_n=c_n/c_n, a_{n-1}=c_{n-1}/c_n, \cdots, a_1=c_1/c_n, a_0=c_0/c_n,$$

并且一定有 $a_n=1$. 很明显，$f(z)$和 $g(z)$的根是完全一致的，所以只须对付后者就可以了.

因此，今后我们只讨论形如

$$f(z)=z^n+a_{n-1}z^{n-1}+\cdots+a_1z+a_0$$

的所谓**首一多项式**，即首项系数为 1 的多项式，其中 z 是复变量，$a_0, a_1, \cdots, a_{n-1}$都是复常数.

库恩算法的目的，就是在复数 z 平面 C 上寻求多项式 $f(z)$ 的根.

我们分三步来叙述这种算法：

1. 半空间 $C\times[-1, +\infty)$ 的一种三角剖分.

首先，把一系列复数 z 平面 C_{-1}, C_0, C_1, C_2, …从下到上排列好，按照图 2.1 的规则，在每个平面上划线，把平面都分割为等腰直角三角形. 注意，在图 2.1 中，每层平面只画出一个象限. C_{-1}和 C_0 这两层线的格距是一样的，但斜线错开. 从 C_0 开始，每上升一层，线距就缩小一半. 算法将用这些越来越细的三角形网格，捕捉多项式的根.

C_{-1}的一个方格与 C_0 的一个方格上下相对，构成一个基本

立方体，方体上下的斜线相错. 这样的基本方体按图 2.2 左的方式，分割成 5 个四面体. C_0 以上，每层上的一个方格与上一层的 4 个方格相对，它们所确定的基本方体，则按图 2.2 右的方式，分割成 14 个四面体. 图 2.2 左画得比较窄，该不会影响你的理解.

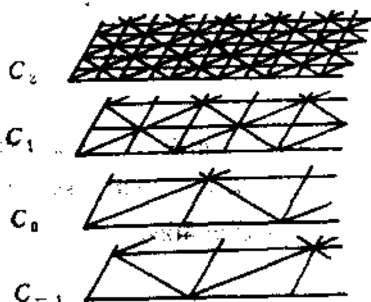


图 2.1

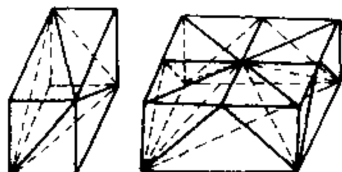


图 2.2

这样一来，半空间 $C\times[-1,+\infty)$ 就被分割为一个个四面体，越往上去，四面体越小.

2. 整数标号法.

记 $w=f(z)$，就可以把多项式 f 看作是从复数 $z=x+iy$ 平面到复数 $w=u+iv$ 平面的变换.

请看图 2.3. 首先，用原点出发的辐角分别为 $\pi/3$，$-\pi/3$ 和 π 的 3 条射线，把复数 w 平面 C' 分成相等的 3 个扇形区域 G_1，G_2 和 G_3. 对于 C_{-1} 平面上的点 z，按照 $w=z^n$ 落在区域 G_1 或 G_2 或 G_3，确定这个点 z 的标号 $l(z)$ 为 1 或 2 或 3. 对于 C_0，C_1，C_2，…平面上的点 z，则按照 $w=f(z)$ 将 z 变换到区域 G_1 或 G_2 或 G_3，确定这个点 z 的标号 $l(z)$ 为 1 或 2 或 3. 因为标号不外乎整数 1 或 2 或 3，这种确定复数 z 的标号的方法，称为**整数标号法**. 要注意的是，C_{-1} 平面的点是由 $w=z^n$ 标号的，而其

它平面 C_0，C_1，C_2，…的点，都由 $w=f(z)$ 标号.

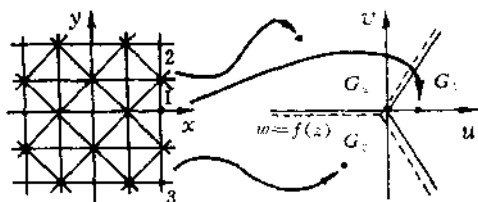


图 2.3

我们已经把各层复平面分割成一个个三角形. 三个顶点的标号都不相同的三角形，即一个顶点标号为 1，一个顶点标号为 2，一个顶点标号为 3 的三角形. 称为**全标三角形**.

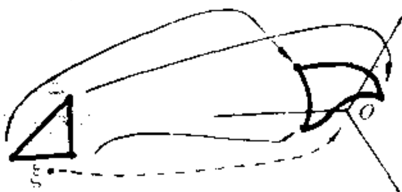


图 2.4

$k\geqslant 0$ 时，C_k 平面上一个全标三角形三个顶点的多项式值 $w=f(z)$ 分别落在 G_1，G_2，G_3 这 3 个区域，它们"包围"着原点. 所以，我们有理由期望全标三角形内或全标三角形附近会有一点 ξ，它将被变换 $w=f(z)$ 送到复数 w 平面 C' 的原点. 这样的一点 ξ，当然应是多项式 $f(z)$ 的根（图 2.4）. 的确，我们将在第 3 节证明这个猜想，我们把它写成定理 1：

定理 1 $k\geqslant 0$ 时，C_k 平面上全标三角形任一顶点与多项式某个根的距离不超过

$$\sqrt{2}(1+3n/4)d/2^k,$$

这里 n 是多项式的阶数，而 d 是 C_0 平面上等腰直角三角形的直角边长.

因为 n 和 d 都是固定的，所以增大 k 就可以使定理中的数变得任意小，也就是说使 C_k 平面上的全标三角形都离多项式的根很近. 因此，只要能在足够高的层次 C_k 上找到全标三角形，多项式的根的近似值也就找到了. 这就是下面要做的. 为了方便起见，我们确定 $d=1$.

我们将在本章 § 3 中证明上述定理 1.

3. 同标号顶替算法.

取 m 为不小于 $3(1+\sqrt{2})n/4\pi$ 的最小整数. 在 C_{-1} 平面上，确定以原点为中心的边长为 $2m$ 的方块 Q_n. 图 2.5 是 $n=3$ 的例子，这时容易算出 $m=2$.

图 2.5

方块 Q_n 的边沿 ∂Q_n 上有 $8m$ 个格点和 $8m$ 条棱. C_{-1} 平面的点是按照幂函数 $w=z^n$ 来标号的，而 $w=z^n$ 是一个十分规矩的多项式. 我们在第 2 节将证明，按照逆时针方向，∂Q_n 上格点的标号的变化，在 1—2—3—1…循环中，只允许踏步或循序渐进，不允许跳跃，即不允许从 1 跳到 3，从 2 跳到 1，从 3 跳到 2. 进而可以证明，∂Q_n 的 $8m$ 条棱中，正好有 n 条标号先 1 后 2 的 (1，2) 棱，没有标号先 2 后 1 的 (2，1) 棱.

计算就从这 n 条 (1，2) 棱开始.

设想从每个 (1，2) 棱出发，各有一条生长着的曲线向 Q_n 里钻，按照“遇到有 1，2 两个标号的所谓全标棱就穿过去”的

规则前进．可以断言，每条曲线总是可以很快到达一个全标三角形．道理很简单：曲线前进的时候，在越过每条全标棱的当时，总是标号 1 的顶点在左，标号 2 的顶点在右．曲线进入某个三角形时，这个三角形就已有一条全标棱是曲线的进口．如果另一顶点标号为 3，那末曲线已找到全标三角形；如果另一顶点标号不是 3，那末它一定和进口两顶点之一配成一条新的全标棱，而曲线就要穿越这个出口全标棱而去．所以，只要未找到全标三角形，曲线就要继续前进．容易理解（请读者自行证明！），Q_n 中的每个三角形顶多允许曲线通过一次，而 Q_n 中三角形数目 $2\cdot(2m)^2$ 有限，所以如果曲线一直找不到全标三角形，就必须穿越 ∂Q_n 离开 Q_n．在穿越 ∂Q_n 离开 Q_n 的当时，标号 1 在左，标号 2 在右，按逆时针方向看，就应该是 ∂Q_n 上的一个（2，1）棱．但是我们已经说过，∂Q_n 上没有（2，1）棱，矛盾．这就说明了，每条曲线必定在有限步内找到 Q_n 中的一个全标三角形．

找到全标三角形后，曲线开始向空中伸延，规则是："遇到全标三角形就穿过去"．穿过一个全标三角形，就进入一个四面体．四面体有 4 个顶点，进口三角形 3 个顶点的标号分别是 1、2 和 3，不论第 4 个顶点的标号是 1 或 2 或 3，总是和前 3 个顶点中的一个标号相同．对于标号相同的顶点，实行以新代旧的同标号顶替，就得到一个新的全标三角形，成为曲线从这个四面体出来的出口，所以曲线在空中伸延，在带有全标三角形端面的四面体中穿行的过程永远不会终止．

容易理解（也请读者自行论证），每个四面体顶多允许曲线穿过一次．根据 z 比较大的时候，首一多项式

$$f(z)=z^n+a_{n-1}z^{n-1}+\cdots+a_1z+a_0$$
$$=z^n(1+a_{n-1}/z+\cdots+a_1/z^{n-1}+a_0/z^n)$$

的行为都和幂函数 z^n 差不多，我们在第 2 节将证明，存在一个正数 R，使得在想象的以原点为中心 R 为半径的大圆筒外，没有全标三角形. 按照规则，曲线只能在带全标三角形的四面体中穿行，所以不能跑出大圆筒. 大圆筒在任何有限高度下的四面体数目是有限的. 既然曲线要无休止地穿行下去，每次都穿过一个新的四面体，它就必须越走越高. 但是定理 1 后面的分析告诉我们，越高的全标三角形离多项式的根越近，要怎么近有怎么近. 这就保证了从方块边沿 ∂Q_n 上 n 条 (1, 2) 棱出发的 n 条曲线，可以无限地越来越接近多项式 $f(z)$ 的 n 个根.

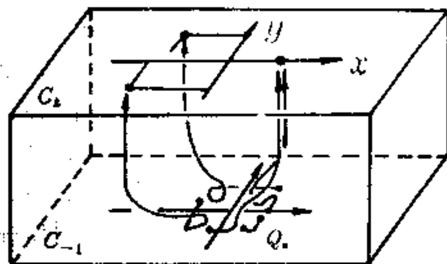


图 2.6

真是一幅神奇的动画：培养皿 Q_n 边上栽着的 n 个幼芽，在一个看不见的大圆筒的屏蔽之下，经过短暂的左右搜索，就不断上攀，正好把多项式的 n 个根一一指示出来.

细心的读者记得，在第一章讲对分区间迭代法时用过同符号顶替. 现在库恩算法用的是同标号顶替. 它们有什么联系吗？的确，它们的思想是一致的，都是要把原点捕捉住. 在实数轴的一维情形，一正一负两个点就把原点卡住了，比较容易. 所以我们用同符号顶替，使函数在区间两端的值总是异号. 在复平面的二维的情形，本来靠几个点是围不住原点的，而是需要

一个圈．库恩算法之巧妙，就是只用三个点，同样达到捕捉原点的目的．三个点如何把原点捉住，这是需要论证的，下一节就专门做这个论证．但是首先这三点要分布得好，要位于图 2.3 的三个不同的标号扇形．为了保持每个标号扇形中恰有一个点，所以用同标号顶替．

本节库恩算法的叙述和下节库恩算法的论证中，使用了"进口""出口"分析．早在库恩论文之前十年，旅美中国数学家樊畿就在美国《组合数学》杂志上发表论文使用进口出口分析．樊的论文说：设想有一座大房子，里面有许多小房间，如果每个房间顶多只有两个门，整个大房子只有一个对外的门，那么大房子里一定有只有一个门的所谓"好房间"．为了论证，樊规定了一种"走法"：从大房子唯一的门进去，每个门只许走过一次，就一定走到好房间．为什么？因为每个房间顶多两个门，所以顶多进一次出一次，穿过之后就不能再回来．如果一直找不到好房间，而大房子内房间数目有限，那么最后只能走出大房子．可是根据假设，大房子只有一个已经用作进口的门，没有第二个门，又要出来，又没有出口．这就导致矛盾．所以，按照樊的"走法"，一定可以找到好房间．

上一章讲过，反证法通常不是构造性的，但樊畿设计了他的"走法"，在此基础上的反证法论证，就是构造性的了，因为他告诉你具体的走法，保证你能找到好房间．

在樊的论文以后，进口出口分析就风行一时，体现了组合讨论的魅力．樊的"走法"，就是人们所讲的"算法"．至于库恩算法，则又进了一步：房间数目无限．这时，全标三角形就是门，全标三角形越小，它所在的"房间"就越好．

§2 可行性和收敛性的论证

这一节我们要论证库恩算法为什么会成功，这包含两方面的内容.

一是可行性的论证，即说明为什么方块 Q_n 的边界 ∂Q_n 上正好有 n 个（1，2）棱，没有（2，1）棱，并且从每个（1，2）棱出发的曲线，都将无休止地伸延，不会发生交叉和打圈这样迷失方向的情形.

二是收敛性的论证，主要是说明为什么每条曲线都要不断向上攀登. 这样，结合待证的定理1，就知道每条曲线正好捕捉住一个根.

我们分几个命题完成这些论证.

命题1 按逆时针方向，∂Q_n 上原点的标号，只能有踏步的 $1\to1$，$2\to2$，$3\to3$ 和渐进的 $1\to2$，$2\to3$，$3\to1$ 六种情况. 所以，∂Q_n 上正好有 n 条（1，2）棱，没有（2，1）棱.

略证 当 m 较大时，方块边界上每条棱 $z'z''$ 离原点就比较远，它们对原点的张角 $|\arg(z''/z')|$ 就比较小. 方块 Q_n 所在的 C_{-1} 平面的点是按照 $w=z^n$ 标号的，而幂函数 $w=z^n$ 把 C 平面上的正方形边界 ∂Q_n 十分对称地在 C' 平面上绕原点 n 圈. 因为 ∂Q_n 上每条棱对原点张角 $|\arg(z''/z')|$ 比较小，那末这条棱作 n 次幂后在 w 平面 C' 上对原点的张角 $|\arg(z''^n/z'^n)|=n|\arg(z''/z')|$ 也就比较小. 微分学的简单极值计算表明，$m\geqslant 3n/2\pi$ 时，就总有 $n|\arg(z''/z')|<2\pi/3$. 复数运算告诉我们，n 次幂的辐角等于辐角的 n 倍. 所以，∂Q_n 上每条棱作 n 次幂后仍不能一下子跨越确定标号的整个扇形区域，所以，∂Q_n 上格点

的标号，只能踏步或渐进. 再注意 ∂Q_n 的像在 w 平面 C' 上正好 n 次由 G_1 进入 G_2，就知道 ∂Q_n 上正好有 n 条 (1，2) 棱. 最后，如果有 (2，1) 棱的话，就是说这条棱的 $w=z^n$ 像从 G_2 跨越整个区域 G_3 进入 G_1. 而这是不可能的，所以(2,1) 棱不会出现.

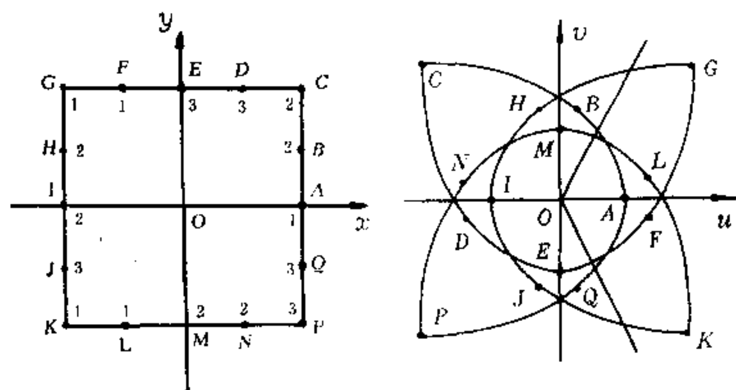


图 2. 7

因为在算法中取 m 为不小于 $3(1+\sqrt{2})n/4\pi$ 的最小整数，当然符合 $m \geqslant 3n/2\pi$ 的要求. 这样，n 条求根曲线的出发点就有了着落. 余下只要说明，这些曲线不会交叉分叉，不会打圈.

求根曲线先是在 C_{-1} 平面的三角形中穿行往 Q_n 里走，规则是遇到有 1、2 两个标号的棱就穿过去. 我们把 C_{-1} 平面上有 1、2 两个标号的三角形叫做**通道三角形**. 很明显，求根曲线走过的三角形，都是通道三角形. 在 C_{-1} 平面上找到全标三角形以后，求根曲线就往空中的四面体中穿行，规则是遇到全标三角形就穿过去. 我们把空中有全标三角形端面的四面体叫做**通道四面体**. 很明显，求根曲线走过的四面体，都是通道四面体.

命题 2 每个通道三角形和通道四面体都正好有一对门，一个是进口，一个是出口.

证明 先看四面体. 既然是通道四面体，就已经有一个面是全标三角形，这已是一个求根曲线可以通过的“门”. 四面体的 3 个顶点已标号 1、2、3，那末无论第 4 个顶点的标号是什么，这个四面体就一定还有并且只还有一个面是全标三角形（请读者多画几张四面体草图弄清这个道理），所以通道四面体都恰有一对门.

再看通道三角形，它已有一条棱标号 1、2，是求根曲线可以通过的门. 如果另一顶点标号是 1 或者 2，那末三条棱中必有两条标号 1、2 的棱，而另一棱两端标号相同，不成为门（也请画图分析），所以该三角形正好有一对门. 如果另一顶点的标号是 3，那末这是一种特殊的通道三角形，担负着从 C_{-1} 平面到空中的转折的任务. 把标号 1、2 的棱作为曲线进入三角形的门，把整个三角形（它是全标的！）作为曲线转入四面体的门，所以也是正好有一对门.

把通道三角形和通道四面体看作是“人”，两个门看作是一双手，这样，求根曲线都是由许多“人”手拉手连起来的. 有了这个比喻，就容易理解，如果曲线会分叉交叉或打圈，就一定要有一些三只手的“人”，与前面证明的每“人”正好有一双手矛盾.

至此，算法的可行性证明完成.

为了证明算法是收敛的，我们需要下面的定理. 刚才命题 2 的证明具有组合的和图论的风格，下面定理 2 的证明却是初等数学的细致推导. 证明的指导思想，却和命题 1 一致.

定理 2 设 $|a_0|$，$|a_1|$，…，$|a_{n-1}|$ 中的最大者记为 a，记 $3\sqrt{2}(2+\pi)n/4\pi$ 和 $1+na/(n-1)$ 中的最大者为 r，再令 $R=$

$r+\sqrt{2}$，那么全标三角形和原点的距离都小于 R.

证明 首先我们注意，若复数 w 符合 $|w|<1$，就必有 $|\arg(1+w)|\leqslant(\pi/2)|w|$. 道理很简单：画图就知 $|\arg(1+w)|\leqslant\arcsin|w|\leqslant(\pi/2)\sin(\arcsin|w|)=(\pi/2)|w|$.

因为三角形的棱都不超过 $\sqrt{2}$，若三角形有一顶点距原点超过 R，整个三角形就都在距原点超过 r 的地方. 下面只须证明，这样的三角形一定不是全标三角形.

$C\times[-1,+\infty)$ 中的顶点都可以表示成 (z,k)，k 是不小于 -1 的整数. 设 (z_1,k_1)，(z_2,k_2) 是上述三角形的两个顶点，若 k_1,k_2 都大于 -1，则两点都由 $w=f(z)$ 标号. 将 $f(z)$ 改写成

$f(z)=z^n(1+a_{n-1}/z+\cdots+a_1/z^{n-1}+a_0/z^n)=z^n(1+g(z))$. 因为 $|z_1|>r$，$|z_2|>r$，就有

$$
\begin{aligned}
|g(z_2)|&\leqslant|a_{n-1}|/r+\cdots+|a_1|/r^{n-1}+|a_0|/r^n\\
&\leqslant a(1/r+\cdots+1/r^{n-1}+1/r^n)\\
&<a/(r-1)\leqslant(n-1)/n,
\end{aligned}
$$

$$
\begin{aligned}
&|g(z_1)-g(z_2)|\\
&\quad\leqslant|a_{n-1}|\cdot|1/z_1-1/z_2|+\cdots+|a_0|\cdot|1/z_1^n-1/z_2^n|\\
&\quad\leqslant a|z_1-z_2|(1/r^2+2/r^3+\cdots+n/r^{n+1})\\
&\quad<\sqrt{2}\,a/(r-1)^2\\
&\quad\leqslant\sqrt{2}(n-1)/n(r-1)\\
&\quad\leqslant\sqrt{2}(n-1)4\pi/3n\sqrt{2}(2+\pi)(n-1)\\
&\quad=4\pi/3(2+\pi)n,
\end{aligned}
$$

$$\left|\frac{g(z_1)-g(z_2)}{1+g(z_2)}\right|\leqslant\frac{4\pi}{3(2+\pi)n}\left(1-\frac{n-1}{n}\right)=\frac{4\pi}{3(2+\pi)}<1.$$

于是，记 z_1 和 z_2 在 w 平面 C' 的象为 w_1 和 w_2，有

$$\left|\arg\frac{w_1}{w_2}\right|=\left|\arg\frac{f(z_1)}{f(z_2)}\right|=\left|\arg\frac{z_1^n(1+g(z_1))}{z_2^n(1+g(z_2))}\right|$$

$$\leqslant\left|\arg\frac{z_1^n}{z_2^n}\right|+\left|\arg\left(1+\frac{g(z_1)-g(z_2)}{1+g(z_2)}\right)\right|$$
$$\leqslant n\cdot\left|\arg\frac{z_1}{z_2}\right|+\frac{\pi}{2}\cdot\left|\frac{g(z_1)-g(z_2)}{1+g(z_2)}\right|$$
$$<\frac{n\sqrt{2}}{3\sqrt{2}(2+\pi)n/4\pi}+\frac{\pi}{2}\cdot\frac{4\pi}{3(2+\pi)}=\frac{2}{3}\pi.$$

当 k_1 和 k_2 有一个是 -1 或两个都是 -1 时，更容易证明 $|\arg(w_1/w_2)|<2\pi/3$，因为可用 z^n 代 $f(z)$.

既然三角形 3 个顶点在 C' 平面上的象，两两之间对原点张开的角度都小于 $2\pi/3$，那末 3 个顶点的象不可能分别位于 G_1、G_2、和 G_3. 这就证明了：有一个顶点距原点不小于 R 的三角形，不会是全标三角形.

有一个面是全标三角形才是通道四面体，所以通道四面体都在以原点为中心 R 为半径的大圆筒内. 这样一来，既然求根曲线要无休止地穿行下去，每个四面体顶多允许它通过一次，所以曲线一定要不断上升. 结合下节将证的定理 1，就知道 n 条求根曲线正好把多项式的 n 个根抓住.

这一节的 3 个证明代表 3 种不同的风格，其中命题 1 的证明突出主要思想，省略了一些技术细节. 我们不希望读者只注重细节，最重要的是理解算法的全局. 如果你能想到下面的问题，就说明你对算法的全局把握得很好.

我们说过，求根曲线在 Q_n 内找到全标三角形后，就开始往空中伸延. 那么，如果曲线转了一阵又回到 C_{-1} 平面上来怎么办？你想到过这个问题吗？不解决这个问题，算法就会乱套.

首先要注意，曲线在 C_{-1} 平面上伸延时，总是标号 1 在左标号 2 在右，而曲线在四面体中穿行时，总是遵守“右手法则”，即看到逆时针标号 1、2、3 的全标三角形就作为进口钻进去，看到顺时针标号 1、2、3 的全标三角形就作为出口钻出去. 所以，

如果曲线走了一阵又回到 C_{-1}平面上来，一定到达顺时针标号的 1、2、3 全标三角形.

出现这种情形时，曲线当然不能穿到C_{-1}平面下面去，因为那里没有四面体. 这时，我们**规定**：曲线看准标号 1 在左标号 2 在右的棱穿出去，再次在 C_{-1}平面上匍匐前进，待走到新的全标三角形时再抬头向空中发展. 新找到的全标三角形一定是逆时针标号 1、2、3 的三角形.

运用与命题 1 同样的方法可以证明，当 $m\geqslant 3(1+\sqrt{2})n/4\pi$ 时，Q_n 外就不会有全标三角形. 这个数比命题 1 本来的 $3n/2\pi$ 大，是因为 ∂Q_n 上的棱都是长度为 1 的棱，而论及三角形时，就要考虑长度为 $\sqrt{2}$ 的斜边.

这样一来，曲线回到 C_{-1}平面的话，就必须回到 Q_n 里面. Q_n 里三角形数目很有限，所以曲线只能回头很少次，就一定从此永远在空中的四面体中穿行.

作为算法的一个简单的练习，请读者看图 2.5，找出可以让曲线回到 C_{-1}平面来的全标三角形，并且从这种三角形出发匍匐前进，找到新的使曲线再度升空的全标三角形.

可能有读者埋怨，为什么这么晚才告诉如何对付曲线回头的情形？其实，如果一开始就左左右右都详说，恐怕不利于读者迅速抓住算法的要点. 做研究工作也是这样，先想出曲线如何穿行的方法，后来发现曲线可能回到 C_{-1}来，再想办法对付这种小麻烦. 没有什么圣人可以一下子把所有可能的情形都想得十分仔细.

§3 全标三角形与根的距离

这一节我们证明 §1 中叙述的定理 1，它说明全标三角形与

多项式的根的距离．因为规定 C_{-1} 和 C_0 平面的线距 $d=1$，所以定理可以写成：

定理1 $k\geqslant 0$ 时，C_k 平面上全标三角形任一顶点与多项式某个根的距离不超过

$$\sqrt{2}(1+3n/4)/2^k.$$

证明 因为 C_k 平面上三角形任两顶点的距离不超过 $\sqrt{2}/2^k$，我们只要证明全标三角形有一个顶点与多项式的某个根的距离不超过 $3n\delta/4$，其中 δ 就是三角形斜边长 $\sqrt{2}/2^k$.

$n=1$ 没什么值得讨论（为什么?），下面只考虑 $n>1$ 的情形．改写多项式为

$$f(z)=(z-\xi_1)(z-\xi_2)\cdots(z-\xi_n),$$

这里 $\xi_1,\ \xi_2,\ \cdots,\ \xi_n$ 是多项式的 n 个根，由代数基本定理保证.

若定理不真，C_k 平面上某个全标三角形 $\langle z_1,\ z_2,\ z_3\rangle$ 各顶点与多项式所有根的距离都大于 $3n\delta/4$，即对所有 $i=1,2,3$ 和 $j=1,\ \cdots,\ n$，都有 $|z_i-\xi_j|>3n\delta/4$．于是，对 $j=1,\ \cdots,\ n$，

$$\left|\frac{z_2-z_1}{z_1-\xi_j}\right|<\frac{\delta}{3n\delta/4}=\frac{4}{3n}<1,$$

所以

$$\begin{aligned}
\left|\arg\frac{w_2}{w_1}\right| &= \left|\arg\frac{f(z_2)}{f(z_1)}\right| = \left|\arg\frac{(z_2-\xi_1)\cdots(z_2-\xi_n)}{(z_1-\xi_1)\cdots(z_1-\xi_n)}\right| \\
&\leqslant \left|\arg\frac{z_2-\xi_1}{z_1-\xi_1}\right|+\cdots+\left|\arg\frac{z_2-\xi_n}{z_1-\xi_n}\right| \\
&= \left|\arg\left(1+\frac{z_2-z_1}{z_1-\xi_1}\right)\right|+\cdots+\left|\arg\left(1+\frac{z_2-z_1}{z_1-\xi_n}\right)\right| \\
&\leqslant \frac{\pi}{2}\cdot\left|\frac{z_2-z_1}{z_1-\xi_1}\right|+\cdots+\frac{\pi}{2}\cdot\left|\frac{z_2-z_1}{z_1-\xi_n}\right| \\
&< n\cdot\frac{\pi}{2}\cdot\frac{4}{3n}=\frac{2}{3}\pi,
\end{aligned}$$

其中倒数第2个不等号是因为 $|w|<1$ 时一定成立 $|\arg(1+w)|$

$\leqslant(\pi/2)|w|$.

同理可证：

$$|\arg(w_3/w_2)|=|\arg(f(z_3)/f(z_2))|<2\pi/3,$$

$$|\arg(w_1/w_3)|=|\arg(f(z_1)/f(z_3))|<2\pi/3.$$

这说明三角形三顶点在 C_{-1} 平面上的象，两两对原点的张角都小于 $2\pi/3$，所以它们不可能一个在 G_1，一个在 G_2，一个在 G_3，这与三角形是全标三角形矛盾.

这就完成了定理的证明.

库恩算法的叙述，需要一定的篇幅. 求根曲线怎样“生长伸延”，则是交给计算机去做的. 把第一节的内容研究清楚，你就可以着手编制库恩算法的计算机程序了.

要编制这个程序，很重要的就是弄清楚三角形和四面体顶点之间的位置关系和坐标关系. 具体地说，当曲线在 Q_n 内匍匐前进时，已知曲线已经到达的通道三角形的 3 个顶点，并且已知哪两个顶点组成出口棱，如何确定下一个通道三角形的另一个顶点的位置和坐标；当曲线在空中穿行时，已知曲线已经到达的通道四面体的 4 个顶点和组成出口全标三角形的 3 个顶点，如何确定下一个通道四面体的另一个顶点的位置和坐标. 这对读者的立体几何概念和空间坐标概念是一种很好的锻炼. 读者应当编制一个不管 n 等于多少都可以用的程序，这就可以一劳永逸地帮助你解决求任何多项式的全部根的问题.

采用库恩算法来求多项式 $f(z)=z^n+a_{n-1}z^{n-1}+\cdots+a_1z+a_0$ 的根的时候，曲线每前进一步，首先要算一个点的标号. 根据标号法，如果这个点不是 C_{-1} 平面上的点，就要算这个点的多项式值. 所以，如何进行多项式计值，值得讲究. 通常使用的多项式计值格式是

$$f(z)=(\cdots((z+a_{n-1})z+a_{n-2})z+\cdots+a_1)z+a_0,$$

它的好处是把乘方运算可能带来的麻烦尽可能避免掉，并且节省运算量.

其它细节问题，读者可自行考虑解决.

下面叙述我们的一些计算结果.

当多项式

$$f(z)=z^6-(4.1+6i)z^5-(4.8-12.3i)z^4+4z^2-(16.4+24i)z-(19.2-49.2i)$$

时，用库恩算法算出 6 个根如下：

$$\xi_1=\ \ 2.1000000+3.0000000i,$$
$$\xi_2=\ \ 1.9999999+2.9999999i,$$
$$\xi_3=-0.9999999+1.0000000i,$$
$$\xi_4=\ \ 0.9999999-0.9999999i,$$
$$\xi_5=-0.9999999-1.0000000i,$$
$$\xi_6=\ \ 1.0000000+0.9999999i.$$

这是一个中学数学决不会遇到的“真正的”复系数多项式.

重要的是记住，多项式求根问题，局限在实数范围内，就不可能彻底解决. 41 阶多项式

$$f(z)=z^{41}+z^3+1$$

是一个实系数多项式，但它的 41 个根中，只有一个是实数根，那就是 $\xi=-0.95248387$. 利用库恩算法，很容易就找到它的全部根：

$$\xi_{1,41}=1.0143046\ \pm 0.080923027i,$$
$$\xi_{2,40}=0.98718386\ \pm 0.24035438i,$$
$$\xi_{3,39}=0.93366404\ \pm 0.39254638i,$$
$$\xi_{4,38}=0.85515802\ \pm 0.53263353i,$$
$$\xi_{5,37}=0.75371952\ \pm 0.65538027i,$$
$$\xi_{6,36}=0.63233982\ \pm 0.75340119i,$$

$$\xi_{7,35} = 0.50756864 \pm 0.81057343i,$$
$$\xi_{8,34} = 0.41715157 \pm 0.87106730i,$$
$$\xi_{9,33} = 0.28981213 \pm 0.94642367i,$$
$$\xi_{10,32} = 0.13916543 \pm 0.99247673i,$$
$$\xi_{11,31} = -0.019728633 \pm 1.0093521i,$$
$$\xi_{12,30} = -0.18020604 \pm 0.99796223i,$$
$$\xi_{13,29} = -0.33698432 \pm 0.95922761i,$$
$$\xi_{14,28} = -0.48528023 \pm 0.89453836i,$$
$$\xi_{15,27} = -0.62067250 \pm 0.80588930i,$$
$$\xi_{16,26} = -0.73910056 \pm 0.69590435i,$$
$$\xi_{17,25} = -0.83686300 \pm 0.56782566i,$$
$$\xi_{18,24} = -0.91051113 \pm 0.42552815i,$$
$$\xi_{19,23} = -0.95633901 \pm 0.27377620i,$$
$$\xi_{20,22} = -0.96814033 \pm 0.12086695i,$$
$$\xi_{21} = -0.95248387 + 0.15894571 \times 10^{-8}i.$$

这 41 个根在 z 平面排成一个以实轴为对称轴的心形，心尖向右，心形的平均半径大约是 1．表列最后一个 ξ_{21}的虚部其实是 0，因为这次计算只用了 8 位数字精度的计算机.

愿意编制库恩算法计算程序的读者，除了重复这两个算例作为检验以外，还可试试把第一章的多项式 $f(z)=z^5-17z+2$ 的所有根算出来. 编程序要花时间，但将是一劳永逸的工作. 另外，喜欢不动点迭代的读者，则可以试试用第一章介绍的不动点迭代方法，把上面 41 阶多项式的那个实根确定下来.

§4 积木结构的计算复杂性讨论

这一节我们讨论库恩算法的效率，具体来说，就是证明多

项式求根的库恩算法，是计算复杂性理论所寻求的多项式时间算法.

算法效率的讨论，换一个角度说，就是计算成本的估计：在用算法解决问题之前，预先估计计算成本是多少.

前面已经讲过，时间是成本的一种尺度，花费时间多的，成本就高. 但计算机是不断升级换代的，速度越来越快，功能越来越强，绝对的时间估计已经难以说明效率或成本. 要说时间，只能说在某种类型的机器上要用多少时间. 本来可以把时间"基本化"，按照解决一个问题机器要做多少次基本运算动作来估计成本，但这样做需要计算机硬件和软件方面的一些知识，未必适合只关心如何使用计算机来解决问题的读者.

面对多项式求根问题，不同的算法可以有不同的尺度. 例如下一章讲的牛顿算法的成本，可以用所需要的牛顿迭代的次数来衡量；库恩算法的成本，则可以用求根曲线穿越 Q_n 内的三角形和空中的四面体的次数来衡量. 重要的是，牛顿迭代次数和库恩曲线穿越次数，都可以进一步转换成统一的多项式计值次数. 事实上，数值地计算一个函数的根，通常需要多次计算函数的值，即在某一点，算出这点的函数值，根据已经算过的点的函数值所提供的信息，指示下一个点应选在什么地方. 对于库恩算法，必须算一个新顶点的标号，曲线才能向前穿越一次. 为了算顶点的标号，就要计算这个顶点的多项式值 $w=f(z)$，或者当这个顶点在 C_{-1}时计算幂函数 $w=z^n$ 值. 曲线在 C_{-1}平面上穿越的是三角形，在空中穿越的才是四面体. 所以，我们可以用曲线穿越四面体的次数，来衡量求根所需要的多项式计值的次数.

事实上，忽略曲线在 Q_n 内匍匐前进的成本很小的一段以后，曲线的继续伸延，可以看作被它穿越的四面体的不断堆砌.

库恩算法的这种积木式的几何结构，是解决成本估计的关键. 因为只有当曲线穿越四面体时，才会需要计算多项式值，而每个四面体顶多允许曲线穿越一次，所以数一下曲线穿越了多少个四面体，就知道做了多少次多项式计值. 这样，计算成本也就知道了.

但是，这样数出来的，只是事后的“成本核算”，而复杂性理论要求的却是事前的“成本估计”. 事前我们不知道曲线将走过哪些四面体，成本怎么估算呢？

解决问题，有赖于下面两个观察：

观察 1 曲线穿越的四面体，都在以原点为中心 R 为半径的大圆筒里.

这不过是 §2 中证明了的定理 2 的一种通俗的表达.

大圆筒的高度是无限的，所含四面体数目也无限. 但是我们记得，越往上，全标三角形越接近多项式的根. 根据全标三角形和多项式根的距离不超过 $\sqrt{2}(1+3n/4)/2^k$ 的定理，如果计算的准确度要求是 ε，我们只要取 K 为不小于 $\log_2(\sqrt{2}(1+3n/4)/\varepsilon)$ 的最小整数，那么曲线到达 C_K 就大功告成了. 这时，全标三角形与多项式根的距离不超过

$$\begin{aligned}&\sqrt{2}(1+3n/4)/2^K\\ \leqslant&\sqrt{2}(1+3n/4)/(\sqrt{2}(1+3n/4)/\varepsilon)=\varepsilon,\end{aligned}$$

说明找到的根误差不超过允许的范围.

在 C_K 以下大圆筒所含的四面体数目是有限的，也很容易算出来，比方说是 M 吧. 这样就可以给出一个成本估计：按照精度要求 ε 算出多项式全部 n 个根的成本，不超过 M 次多项式计值.

我们不把这个 M 算出来，因为它肯定不能令人满意. 大圆筒是上下一般粗的，每上升一层，四面体的数目就增至 4 倍，这

样的指数增长，比梵天宝塔问题中的指数增长还厉害.

求根曲线真的可能跑遍大圆筒吗？

不！

经过认真的研究，我们作出了这个回答. 事实上，定理1告诉我们，当 $k\geqslant 0$ 时，求根曲线与多项式的根的距离不超过 $\sqrt{2}(1+3n/4)/2^k$，这个数上升一层就要缩小一半. 多项式有 n 个根（其中有些可能相重），曲线只能在以这 n 个根为中心的 $\sqrt{2}(1+3n/4)/2^k$ 为半径的圆柱内生长. 由于 $1/2^k$ 的作用，每向上一层，圆柱直径减半，所以这些圆柱，实际上是 n 座以多项式的根为轴线的圆柱阶梯. 这样，综合定理2和定理1，我们有了更深刻的观察：

观察 2 求根曲线可能走到的地方，由一个半径为 R 的底座圆盘和 n 座半径不断对半缩小的圆柱阶梯组成.

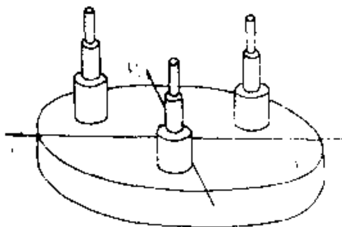


图 2·8

现在，每节小圆柱里面的四面体的数目是一样的. 把 C_K 以下底座圆盘和圆柱阶梯中四面体的数目算出来，就有可能取得成功.

底座圆盘的体积是 πR^2. 每个基本方体体积为1，被分成5个四面体（图2.2左），由此可知，底座圆盘里四面体的总数是 $5\pi R^2$.

C_k 上面一节小圆柱的体积是 $\pi(\sqrt{2}(1+3n/4)2^{-k})^2=2\pi(1+3n/4)^2 2^{-2k}$，而 C_K 上基本方体的体积是 $(1/2^k)^2=2^{-2k}$，分成14个四面体，所以每节小圆柱中四面体的数目是 $28\pi(1+3n/4)^2$.

圆柱阶梯的高度是 K，所以 n 座圆柱阶梯包含的四面体总数是 $28\pi n(1+3n/4)^2K$.

综上所述，求根曲线可能走到的地方的四面体总数，不超过

$$5\pi R^2+28\pi n(1+3n/4)^2K.$$

现在设 $a=1$，即设多项式的系数的模都不超过 1，把上述总数估算一下. 首先，R 是 $3\sqrt{2}(2+\pi)n/4\pi$ 和 $1+an/(n-1)$ 中的大者再加 $\sqrt{2}$，这时 $1+an/(n-1)<3$，所以

$$\begin{aligned}R &= \sqrt{2}+3\sqrt{2}(2+\pi)n/4\pi \\ &<2+3\times1.42\times5.2n/(4\times3.1)<2(n+1),\end{aligned}$$

可知底盘中四面体总数不超过

$$5\times3.2\times(2(n+1))^2=64(n+1)^2.$$

一节小圆柱中四面体数目不超过

$$\begin{aligned}&28\pi(1+3n/4)^2<28\times3.2(1+3n/4)^2<90(1+3n/4)^2\\ &=90\times\frac{9}{16}\left(n+\frac{4}{3}\right)^2=90\times\frac{9}{16}\left(n^2+\frac{8}{3}n+\frac{16}{9}\right)\\ &<90\times\frac{9}{16}\left(n^2+\frac{1}{3}n^2+\frac{8}{3}n+\frac{16}{9}-1\right)\\ &=90\times\frac{9}{16}\times\frac{4}{3}\left(n^2+2n+\frac{21}{36}\right)\\ &<90\times\frac{3}{4}(n+1)^2<68(n+1)^2,\end{aligned}$$

共有 n 个圆柱阶梯，每个 K 节，合起来，总数

$$\begin{aligned}&5\pi R^2+28\pi n(1+3n/4)^2K\\ &<64(n+1)^2+68(n+1)^2nK=(n+1)^2(64+68nK)\\ &<(n+1)^2 68(n+1)K=68(n+1)^3K,\end{aligned}$$

其中 K 是不小于 $\log_2(\sqrt{2}(1+3n/4)/\varepsilon)$ 的最小整数.

计算复杂性理论主要看 n 增加得比较大时计算成本如何增加，而 n 比较大时，易知上述

$$68(n+1)^3K<100n^3\log_2(n/\varepsilon).$$

归纳上述推导，我们得到下面的定理：

定理 3 设 $a=1$，则利用库恩算法按照误差不超过 ε 的要求算出多项式全部 n 个根所需要的多项式计值次数不超过 $100n^3\log_2(n/\varepsilon)$.

前面说过，多项式计值次数反映多项式求根的成本. 对于固定的精度要求 ε，成本随 n 增加的速度不超过 $100n^3\log_2 n \ll 100n^4$. 这说明，库恩多项式求根算法，确实是多项式时间算法，是计算复杂性理论所说的好的算法. 需要补充一句的是：不论把 a 取得多么大，都得到多项式时间的结果.

在下一章，我们会谈一点和定理 3 有关的故事. 从这两页的推导，读者可能觉得有点粗糙，不够细致. 是的，我们在估算多项式计值次数时，匏得不紧. 为什么从 68 就放大到 100 呢？也许还可以小一些，但是，重要的是我们已经成功地论证了库恩算法是一种多项式时间算法，这是决定性的一步.

定理 3 中的 $100n^3\log_2(n/\varepsilon)$ 是不是还可以缩小？完全有这个可能. 其实，即使图 2.8 所表示的塔形地方，也不是被通道四面体填满的. 允许求根曲线穿越的通道四面体，只占了塔形体积的一小部分. 在 C_K 下面，究竟通道四面体的数目是多少，仍然很值得研究. 特别地，如果求根曲线是单调上升的，即一直上升总不下降，通道四面体的数目就一定更小.

读者如果对上述问题有心得，请一定要珍惜. 它们都是有待进一步的科学研究去解决的问题.

第三章　斯梅尔对牛顿算法的研究

牛顿（I. Newton）是 17 世纪中到 18 世纪 20 年代的大科学家.他不但在物理学方面有牛顿三大定律这样伟大的发现，而且对近代数学的发展也作出重大贡献. 牛顿是英国人，他和同时代的德国数学家莱伯尼兹（G. Leibniz），被公认为微积分的创始人.

在解方程方面，他提出了方程求根的一种迭代方法，被后人称为牛顿算法. 三百年来，人们一直使用牛顿算法，改善牛顿算法，不断推广牛顿算法的应用范围. 牛顿算法，可以说是数值计算方面的最有影响的计算方法.

数值计算的复杂性理论，是新兴的科学研究. 它要站得住脚，要显示力量，自然就希望在牛顿算法上做出样子来. 美国加利福尼亚大学伯克利校教授斯梅尔（S. Smale）这样做了，并且取得很大成功，成为数值计算复杂性理论的划时代的工作.

这一章，就介绍斯梅尔的这项研究.

§1　多项式求根的牛顿算法

牛顿算法既可以对付实变量，也可以对付复变量．从易到难，我们先从读者比较习惯的实变量情形讲起．

数学式子写下来，总难免使人感觉枯燥．牛顿算法也是这样．所以，开始时我们强调从几何上学习牛顿算法，学会按照牛顿算法的思想，依靠画图解决问题．读者将体会到，画图帮助思考，是学习数学的好方法．

在实变量的情形，一个函数 $y=f(x)$，通常可以用 x-y 平面上的一条曲线来表示．中学里讲过圆的切线．现在我们说说，曲线在一点的切线是怎么回事．

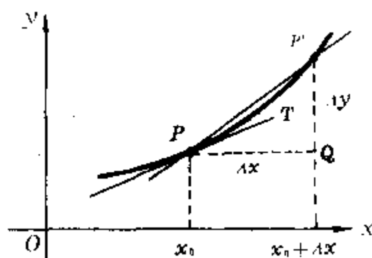


图 3.1

图 3.1 中的粗实线，表示函数 $y=f(x)$ 的图象的一部分．设 P 是曲线上的一点．在曲线上取邻近 P 的另一点 P'，作经过 P 和 P' 的割线．当点 P' 沿着曲线 $y=f(x)$ 趋近到 P 的时候，割线 PP' 的极限位置 PT，叫做曲线 $y=f(x)$ 在点 P 的**切线**．

如果曲线是圆的话，这样得到的切线也和以前所说的圆的切线一样，与圆的半径垂直．

值得注意的是，图 3.1 中的点 P'，既可取在点 P 的右邻，也可取在点 P 的左邻．从图象分析可以知道，只要曲线在点 P

附近是光滑的，曲线在 P 点就有（唯一的）切线. 如果曲线在 P 点断开，曲线在 P 点就没有切线. 如果曲线在 P 点虽然不断开，但是折起一个角，那末把 P' 取在右侧时割线的极限和把 P' 取在左侧时割线的极限不一样. 这时，也说曲线在 P 点没有切线. 例如，心形曲线在心尖那里就没有切线.

在曲线的任一点附近，这一点处的切线是最接近曲线的直线. 曲线比较复杂，计算起来麻烦. 切线是直线，计算起来就比较容易. 用切线代替曲线进行计算，就是牛顿算法的基本思想，我们以此来说明函数求根的牛顿算法.

设 $f(x)$ 是一个函数. 使得函数值等于 0 的点，通常叫做函数的**零点**，即：说 ξ 是 $f(x)$ 的零点，就是指 $f(\xi)=0$. 但是为了与本书主要讨论的多项式的说法一致，我们也说 ξ 是函数 $f(x)$ 的根. 简言之，在本书中，根就是零点，零点就是根.

请看图 3.2，粗实线代表函数 $y=f(x)$. 如果我们已经知道曲线上的一点 $(x_0, f(x_0))$，那末从这点出发沿曲线走，应该可以走到表示函数的根的一点 $(x^*, 0)$. 曲线是比较复杂的，沿曲线走说说容易，做起来就不容易. 但是既然切线是最贴近曲线的直线，为什么不试试沿切线走，看能不能找到函数的根？从已知点出发沿切线走到 x 轴上，这就是牛顿方法！

这样，我们从 $(x_0, f(x_0))$ 出发，沿曲线在这点的切线，走到了 x 轴上的一点 $(x_1, 0)$. x_1 还不是函数的根，这并不奇怪，因为切线毕竟不是曲线本身，但是从图上可以看出，比起原来的 x_0 来，x_1 离 x^* 近得多了. 再从 $(x_1, f(x_1))$ 出发，沿曲线在 $(x_1, f(x_1))$ 这点的切线走到 x 轴上的 $(x_2, 0)$，就靠 x^* 更近了. 如果你还不满意，那末还可以从 $(x_2, f(x_2))$ 出发沿曲线的切线走到 x 轴上的 $(x_3, 0)$，这个 x_3 恐怕要用放大镜才能画在图上. 这样，x_0，x_1，x_2，…一点点走过去，很快就会到达

函数的根 x^*.

读者可以把 x_0 放在图上 A 或 B 的位置，采用牛顿的沿切线走的方法，看看能不能找到函数的根. 牛顿方法的要义是沿切线走，所以又叫做切线法.

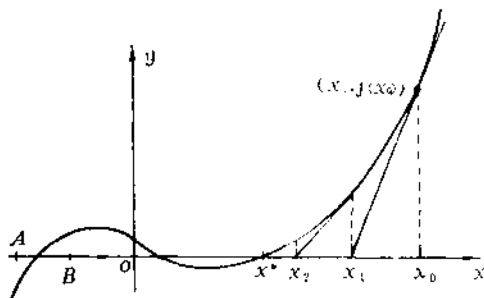


图 3-2

从 x_0 到 x_1 的几何含义已经清楚，现在我们转而看看计算式子应当怎样写. 几何图形对理解方法很有好处，具体计算数据却还得靠计算公式提供. 何况，计算机从根本上说是只会做代数运算的.

看图 3.2，从 x_0 到 x_1，哪些东西是已知的？首先，x_0 是知道的，进而 $f(x_0)$ 也就知道了. 三点 $(x_0, 0)$，$(x_0, f(x_0))$ 和 $(x_1, 0)$ 确定一个直角三角形，现在是已知前两点，要算出第三点，这只要知道斜边的斜率就可以做到.

斜边就是曲线在点 $(x_0, f(x_0))$ 处的切线，切线的斜率怎么确定呢？我们回到图 3.1. 切线是割线 PP' 的极限位置. 割线的斜率是差商

$$\frac{\Delta y}{\Delta x}=\frac{f(x_0+\Delta x)-f(x_0)}{\Delta x},$$

即两点函数值之差和两点自变量之差的比. 所以，切线的斜率就是上面这个比式当 Δx 趋于 0 时的极限，这个极限就是微分学里学的函数 $f(x)$ 在 $x=x_0$ 处的**微商**(也称作**导数**),并且记作 $f'(x_0)$. 所以,曲线 $y=f(x)$ 在点$(x_0,f(x_0))$ 处的切线的斜率,就是函数 $f(x)$ 在 $x=x_0$ 处的微商 $f'(x_0)$. 微商就是差商的极限.

这样，在图 3.2 中，直角三角形两条直角边的长分别是 x_0-x_1 和 $f(x_0)-0=f(x_0)$,斜边的斜率是 $f'(x_0)$,所以两直角边的边长关系是

$$f'(x_0)(x_0-x_1)=f(x_0).$$

因为 $f(x)$是已知的,它在 $x=x_0$ 的值 $f(x_0)$和它的微商在 $x=x_0$ 处的值 $f'(x_0)$都算作已知.所以在上式中,只有 x_1 未知,它正是要算的东西,于是我们得到

$$x_1=x_0-f(x_0)/f'(x_0),$$

这就是按照牛顿方法从 x_0 算 x_1 的公式.

同样,从 x_1 计算 x_2 的公式是

$$x_2=x_1-f(x_1)/f'(x_1),$$

归纳起来,可以写出按照牛顿方法从一点计算下一点的公式

$$x_k=x_{k-1}-f(x_{k-1})/f'(x_{k-1}),\quad k=1,2,3\cdots$$

这就是牛顿算法的迭代公式.

本书主要讨论多项式求根的计算复杂性问题.如果你没有学过微分学,不知道一般函数的微商怎么计算,那末我告诉你,多项式的微商却特别容易计算.事实上,$f(z)=a_nz^n+a_{n-1}z^{n-1}+\cdots+a_1z+a_0$ 的微商函数(它也是一个函数!)就是

$$f'(z)=na_nz^{n-1}+(n-1)a_{n-1}z^{n-2}+\cdots+2a_2z+a_1,$$

其规律是：a_kz^k 项变成 ka_kz^{k-1}，即原来 k 次幂的项，系数乘 k，指数减 1. 所以最后两项 a_1z+a_0，在求微商之后变成 $1a_1z^{1-1}+0a_0z^{0-1}=a_1$. 已知一个函数求它的微商函数的运算，叫做求微商运算或求导运算. 只要正确掌握微商概念，上面介绍的多项式求导法则，当多项式是熟悉的实变量多项式时，可以用一个式子

$$
\begin{aligned}
(a_kx^k)' &= \lim_{\Delta x\to 0}\frac{a_k(x+\Delta x)^k-a_kx^k}{(x+\Delta x)-x} \\
&= \lim_{\Delta x\to 0}\frac{a_k(C_k^1x^{k-1}\Delta x+C_k^2x^{k-2}\Delta x^2+\cdots+C_k^k\Delta x^k)}{\Delta x} \\
&= \lim_{\Delta x\to 0}a_k(C_k^1x^{k-1}+C_k^2x^{k-2}\Delta x+\cdots+C_k^k\Delta x^{k-1}) \\
&= C_k^1a_kx^{k-1}=ka_kx^{k-1}
\end{aligned}
$$

来证明. 但是需要指出，当多项式是复变量多项式时，上述多项式求导运算法则仍然成立.

例如，若 $f(x)=x^3+2x^2+10x-20$，那么就得 $f'(x)=3x^2+4x+10$；若 $f(z)=z^{41}+z^3+1$，就得 $f'(z)=41z^{40}+3z^2$.

至此，读者已经会用牛顿算法为多项式求根.

例 1 从 $x_0=1$ 开始，用牛顿算法求实变量多项式 $f(x)=x^3+2x^2+10x-20$ 的根.

这时，迭代公式是

$$
x_k=x_{k-1}-(x_{k-1}^3+2x_{k-1}^2+10x_{k-1}-20)/(3x_{k-1}^2+4x_{k-1}+10),
$$

容易算出

$x_1=1.411764706$,

$x_2=1.369336471$,

$x_3=1.368808189$,

$x_4=1.368808108$,

……

收敛得很快.

例 2 从 $z_0=\cos\frac{\pi}{41}+i\sin\frac{\pi}{41}$ 开始,用牛顿算法求复变量多项式 $f(z)=z^{41}+z^3+1$ 的根.

这时,迭代公式是

$$z_k=z_{k-1}-(z_{k-1}^{41}+z_{k-1}^3+1)/(41z_{k-1}^{40}+3z_{k-1}^2).$$

这种计算本该交给机器去做的,而且复数运算又比实数麻烦,所以我们为了演示的目的,限于利用多项式本身的特点,迭代一次试试.

因为 $z_0=\cos\frac{\pi}{41}+i\sin\frac{\pi}{41}$,所以 $z_0^{41}=-1$. 利用这个特点,我们有

$$\begin{aligned}z_1&=z_0-(z_0^{41}+z_0^3+1)/(41z_0^{40}+3z_0^2)\\&=z_0-z_0^3/(41z_0^{40}+3z_0^2)\\&=z_0-z_0^4/(41z_0^{41}+3z_0^3)\\&=z_0-z_0^4/(3z_0^3-41),\end{aligned}$$

再按照 $\pi/41$ 即 4.390°,可以得到

$$z_1=1.0219+0.08487i.$$

我们在上一章已经求过这个多项式的全部 41 个根,其中一个是 $\xi=1.0143046+0.0809230i$. 现在只迭代一次,z_1 就和这个 ξ 十分接近,说明收敛很快.

§2 牛顿方法什么时候听话

牛顿算法有一些很明显的优点. 首先,牛顿算法的叙述比较简单,一个迭代公式就把算法基本上说清楚了. 只要你会做微分运算——这对于多项式是十分容易的,牛顿算法也就像第

一章讲的不动点迭代算法一样，是一种一学就会的算法. 另外，数学家已经证明，如果你把迭代出发点 z_0 选得离多项式或函数的根 ξ 很近，并且这个根是单根的话，那末牛顿迭代一定会收敛到 ξ，并且一定收敛得很快. 在这种情形下，牛顿算法的收敛速度，比库恩算法快得多.

牛顿算法与库恩算法还有一点不同，就是一次只能算一个根. 从理论上讲，这不是致命的弱点，因为算出多项式 $f(z)$ 的一个根 ξ 以后，用 $(z-\xi)$ 除原来的多项式，得到一个阶数降 1 的多项式，再用牛顿算法求这个新的多项式的根，…，这样一次一次降阶，该可以把原多项式的全部根算出来.

但是理论情形与实际计算之间难免会有差距. 理论推导中的 ξ 是多项式的准确根，但是机器算出来的，一般只是准确根的近似值. 所以在实际计算时，$(z-\xi)$ 这个因式也只是一个近似因式. 一次一次用近似因式除多项式，就会造成误差积累，即误差越来越大. 这就使采用牛顿算法求高阶多项式全部根的计算，有时候会变得不可靠. 相反，库恩算法是用 n 条求根曲线去搜捉多项式的全部 n 个根，n 条曲线可以齐头并进，相互之间绝不干扰，不会造成误差积累. 顺便指出，库恩算法这种 n 条求根曲线可以互不干扰地齐头并进的特性，使它很自然地可以实施为**并行计算**. 并行计算是大大提高机器解题能力的新的计算方式. 未来的第五代计算机，就十分重视并行计算的功能.

牛顿算法要计算函数值和函数的微商的值，而库恩算法完全不必计算微商，这也是二者的不同之处. 所以，两种算法是各有所长.

牛顿算法最叫人头痛之处，就是有时候很不听话. 的确，牛顿算法像第一章讲的不动点迭代方法一样，很容易学，但是计算是否成功，却是没有保证的. 容易学但并不保证成功，这就

令人感觉头痛．反观第二章讲的库恩算法，虽然有时算得慢一些，却是保险成功的算法．

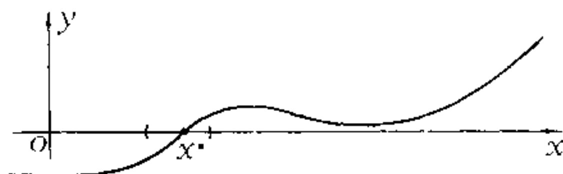


图 3.3

请看图 3.3 的曲线所代表的一个多项式，x^* 是它的一个根．如果从 x^* 附近开始进行牛顿迭代，的确很快收敛到 x^*．但是如果从离 x^* 较远的地方开始进行牛顿迭代，计算或者遇上微商等于 0 就做不下去，或者摇过来晃过去和你捉迷藏就是不收敛．请读者把图 3.3 描下来，实实在在用几何作图的牛顿方法，多做几种迭代试试．

第一章开头介绍不动点迭代方法，只是作为引子，我们并不准备深入讨论那种方法．但是回忆图 1.2 前后的文字，仍然可以从中得到启发．那里，我们曾经以迭代结局为标准来瓜分数轴，把数轴分成大小不等的"地盘"．对于牛顿算法，数学家已经证明只要迭代出发点 z_0 充分靠近多项式的某个单根，计算就会收敛到这个根，并且收敛很快．这么说来，在多项式的每个单根附近，有一个**快速收敛区**这样的"地盘"，如果牛顿迭代的出发点选在这个快速收敛区里，计算就会很快收敛到这个单根上去．在实变量的情形，单根的快速收敛区是包含这个单根的某个开区间（参看图 3.3）．在复变量的情形，单根的快速收敛区是包含这个单根的某片开区域（参看后面的图 3.4）．

既然牛顿算法是如此豪放不羁的方法，我们有必要先把它

的快速收敛区研究一番，因为快速收敛区正是使牛顿算法听话的地盘．研究快速收敛区有若干不同的途径，下面我们只介绍斯梅尔的做法．

从对牛顿算法的初步了解我们也已知道，要使牛顿算法成功，重要的是找到一个足够好的迭代出发点 z_0．迭代出发点也叫做迭代的**初值**．什么叫做足够好的初值呢？斯梅尔提出了**良好初值**的概念：如果 z_0 使得牛顿迭代

$$z_k = z_{k-1} - f(z_{k-1})/f'(z_{k-1})$$

对一切 $k=1,2,3,\cdots$ 都有意义，迭代序列 $z_0,z_1,z_2,\cdots$ 收敛到 $f(z)$ 的一个零点，并且对所有 $k=1,2,3,\cdots$ 都成立

$$|f(z_{k+1})/f(z_k)| < 1/2,$$

就称 z_0 是 $f(z)$ 的一个良好初值．用普通语言写下来，良好初值的意思就是：从这点开始的牛顿迭代可以一直做下去，收敛到函数的一个根，并且每次迭代都使函数值的绝对值衰减一半以上．我们知道，当函数值的绝对值衰减到零时，准确根就找到了，所以，当用牛顿算法为函数求根时，一旦找到一个良好初值，往后就是一马平川，坐等成功的事．

有了明确的概念，接下去就要确立到达良好初值的条件，并设计寻找良好初值的方法．

拿多项式来说，使得多项式 $f(z)$ 的微商函数 $f'(z)$ 的值为零的点，称作多项式 $f(z)$ 的**临界点**，这点的多项式值，称作多项式的**临界值**．读者已经知道 n 阶多项式 $f(z)$ 的微商函数 $f'(z)$ 是 $n-1$ 阶多项式，有 $n-1$ 个根．由此可见，n 阶多项式有 $n-1$ 个临界点（可以相重）．把 n 阶多项式 $f(z)$ 的所有 $n-1$ 个临界值（亦可相重）的绝对值按大小排列起来，取最小的一个，记作 $\rho=\rho(f)$，我们叫做多项式 $f(z)$ 的**表征值**．

经过深入的研究，运用单复变函数论中与单叶函数理论有

关的结果，斯梅尔证明了下面的定理：

定理 1 如果一点 z_0 的多项式值的绝对值小于多项式的表征值的十三分之一，即如果

$$|f(z_0)|<\rho(f)/13,$$

z_0 就是用牛顿算法求多项式 $f(z)$ 的根的一个良好初值.

这个定理的证明比较复杂，难以在本书介绍. 单复变函数，就是一个复变量的函数. "单叶函数理论"则是单复变函数论方面的一项专门研究. 读者也许知道单位圆内单叶函数 $\varphi(z)=z+a_2z^2+a_3z^3+\cdots$ 的系数一定满足 $|a_k|\leqslant k$，$k=2$，3，…的比勃巴赫（Bieberbach）猜想，这个猜想在 1985 年被德布兰吉斯（De Branges）证实. 这些都属于单叶函数理论的核心. 围绕比勃巴赫猜想，有过一些很有趣的科学故事. 当德布兰吉斯宣布他证明成功时，没有人相信他会取得这么伟大的成果. 要知道，半个多世纪以来许多数学家都在啃比勃巴赫猜想. 那真是蚂蚁啃骨头的工作，不但是一个系数一个系数来研究，而且假如对于系数的估计从比如说 $1.08n$ 改进到 $1.07n$，也值得写一篇大论文. 现在，德布兰吉斯这样一位曾经因为发表有错误的论文而被人瞧不大起的人居然一下子彻底证明了比勃巴赫猜想，难怪招来一阵怀疑. 幸亏美国学界有学术休假和学术访问的制度，这位法裔美国数学家在美国被人冷落，就到列宁格勒寻找知音. 苏联数学家们耐心地和他进行了多次讨论，终于弄清了他的思路，看懂了他的证明. 消息反馈到美国，德布兰吉斯才声名大噪.

定理 1 给出了良好初值的一个充分条件，符合这个条件，就一定是良好初值. 这个条件要能够起作用，前提是 $\rho(f)>0$. 按照表征值的定义，$\rho(f)>0$ 正好是多项式没有重根的特征. 所以，定理 1 只能用于没有重根的多项式.

设 $f(z)$是一个没有重根的 n 阶多项式，那末从复数 z 平面到复数 w 平面的变换 $w=f(z)$，在每个根附近差不多就是一个线性变换. 事实上，仍记 $\xi_1,\cdots,\xi_n$ 为 $f(z)$的 n 个根，在 ξ_1 附近我们就有

$$\begin{aligned} f(z) &= [(z-\xi_2)\cdots(z-\xi_n)](z-\xi_1) \\ &\approx [(\xi_1-\xi_2)\cdots(\xi_1-\xi_n)](z-\xi_1), \end{aligned}$$

最后的方括号只含一非零复常数. 这样，根据中学的复数知识就知道，变换 $w=f(z)$把复数 z 平面 ξ_1 附近的一小片，变成复数 w 平面原点附近的一小片，几乎只做了伸缩和旋转. 在其它单根 $\xi_2,\cdots,\xi_n$ 附近，也是这样.

定理得到的充分条件是 $|f(z_0)|<\rho(f)/13$. 在 w 平面上，$|w|<\rho(f)/13$ 确定以原点为中心、$\rho(f)/13$ 为半径的开圆域. 在 z 平面上，$|f(z)|<\rho(f)/13$ 确定的，就是这个圆域在变换 $w=f(z)$下的原像. 既然该变换在单根附近是近似线性的，就知道条件 $|f(z)|<\rho(f)/13$ 确定的，是 z 平面上以 $f(z)$的 n 个根为"中心"的 n 个近似小圆域. 由此可见，定理 1 的充分条件是在 z 平面上确定以多项式的根为中心的 n 个近似圆形的快速收敛区. 一旦进入了这些快速收敛区，牛顿算法的收敛速度将会很快.

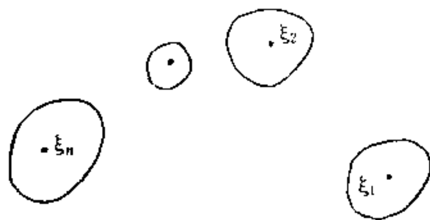


图 3.4

在定理1中，斯梅尔自己用的名称是逼近零点(approximate zeroes)，我们在上面的介绍中，为了使本书的读者看起来通俗一些，改称为良好初值．这点需要提供给查阅文献的读者注意．

§3　概率论定牛顿算法是多项式时间算法

骑手爱烈马，可以说明人们为什么对牛顿算法那么钟爱．要想“烈马”跑得好跑得快，最要紧的是为它找一个好的起跑点．现在，我们考虑如何具体把良好初值找到的问题．

设想把原来牛顿迭代的步长适当缩小，变得精细一些，可望有所帮助．斯梅尔在原来的牛顿迭代公式中引进一个步长参数 h，它符合 $0<h\leqslant 1$．从复数 z 平面的一点 $z(0)$ 开始，归纳地定义

$$z(k)=z(k-1)-h\cdot f(z(k-1))/f'(z(k-1)),\quad k=1,2,\cdots$$

和原来的迭代公式比较，除了 z_k 写成 $z(k)$ 以外，就是作为迭代修正项的第二项前面，多了一个常实数因子 h．我们把上述迭代叫做**参数牛顿迭代**，而把 h 叫做迭代的**步长参数**．很明显，当步长参数 $h=1$ 时，参数牛顿迭代就是原来的牛顿迭代．我们希望，适当选取步长参数 h，经过从初始点 $z(0)$ 开始的若干步参数牛顿迭代，可以达到多项式的一个良好初值(参看图3.5)．

这么一来，$z(0)$ 又是一种初值．$z(0)$ 选得好，可以较快达到一个良好初值，即较快进入快速收敛区．$z(0)$ 选得不好的话，说不定一直不能进入快速收敛区．$z(0)$ 的选择，是因多项式而定的．对于这个多项式是很好的 $z(0)$，对于另一个多项式则可能不好．所以，初始点 $z(0)$ 的好坏，取决于它和具体多项式的关系．

怎样刻划初始点 $z(0)$ 与多项式的关系的好坏呢？斯梅尔引进参数 $\mathscr{K}$，在个别次要的技术性限制之下，定义 $\mathscr{K}$ 为各个临界值与初始点多项式值之比的辐角的绝对值的最小者. 由于这个 $\mathscr{K}$ 是同时依赖于多项式 $f(z)$ 和初始点 $z(0)$ 的，我们可以写 $\mathscr{K}=\mathscr{K}(f,z(0))$. 如果 $\rho(f)$ 和 $\mathscr{K}(f,z(0))$ 都大于 0，初始点 $z(0)$ 对于多项式 $f(z)$ 来说就算不错. 事实上，斯梅尔在引进又一个参数 $\zeta=\zeta(f,z(0))=13|f(z(0))|/\rho(f)$ 之后，证明了下述定理：

定理 2 若多项式 $f(z)$ 和初始点 $z(0)$ 的关系符合 $\rho(f)>0$ 和 $\mathscr{K}(f,z(0))>0$ 的条件，则取

$$h=\frac{\sin(\mathscr{K}/2)}{4(3\sin(\mathscr{K}/2)+\log\zeta)},$$

就可保证从 $z(0)$ 开始的以 h 为步长参数的参数牛顿迭代都有意义，并且顶多经过

$$s=4(3+\log\zeta/\sin(\mathscr{K}/2))^2$$

步参数牛顿迭代，就一定可以到达多项式 $f(z)$ 的一个良好初值.

这个定理的证明也比较复杂，难以在本书叙述. 不过可以指出，在定理 1 的基础上证明定理 2，并不像定理 1 那样需要用到单叶函数理论那样深刻的理论，而是有相当篇幅的关于复数的模和辐角的关系的细致推导. 如果说定理 1 的证明体现一个难字、那么定理 2 的证明则突出一个繁字. 从文献上跟着斯梅尔左转右拐走过长长的路看懂这个证明并不容易，更可想像当年研究出这个定理时的困难. 科学研究不仅要求广博的知识，而且需要艰苦的努力.

但是，广博的知识加艰苦的努力并不等于成功. 只有在创造性思维的驾驭之下，广博的知识才会有用武之地，艰苦的努

力才能结晶出丰硕的果实. 最后得到的定理 3，充分体现了斯梅尔的创造思维.

面对一个业已确定的多项式，可以讨论某个初始点选得好还是不好，这就是前面介绍的内容. 按照这样的思路，本可以这样来讨论牛顿算法的计算复杂性：首先，对任一确定的多项式，判别 z 平面上哪些点可以作为好的初始点，计算好的初始点的集合与全平面的面积比（如果这样做，“面积比”当然要做一些限制）；第二步，把对所有多项式得到的面积比进行适当的平均. 这样得到的平均值，就是参数牛顿迭代成功的概率.

上述想法，可以说是解决问题的一个方案，但是，这个方案行得通行不通，它会引导我们达到什么目标，值得思考一番. 首先看方案的可行性：先给定多项式来判别每个点是不是好的初始点，点这头是无限的，再对所有多项式做平均，多项式这头是无限的. 两头都无限，恐怕很难把握. 再想远一点，即使千辛万苦做下去，并且成功了，也只是平均来说参数牛顿迭代成功的概率是多少这样一个结论. 不解决多项式求根需要做多少次迭代这样的计算复杂性问题.

搞科学研究，不仅要埋头拉车，尤其须抬头看路. 正确的决断，往往是成功的一半. 原来的思路行不通，就应该换一个角度看看. 前面已说清楚，面对一个已确定的多项式，可以讨论某个初始点选得好还是不好；反过来，面对一个已确定的初始点，可以讨论某个多项式（与这个给定的初始点匹配得）好还是不好. 斯梅尔取定 $z(0)=0$，即以不变应万变，总是取原点为参数牛顿迭代的初始点，讨论每个多项式与这个固定的初始点匹配得好不好. 前一章讲过，我们只要讨论首一多项式

$$f(z) = z^n + a_{n-1}z^{n-1} + \cdots + a_1 z + a_0$$

就够了. 一个上述形式的首一多项式，被 n 个复数 a_0，a_1，…，

a_{n-1}完全确定，反过来也一样. 所以我们可以把由所有 n 阶首一多项式组成的空间（读者亦暂可理解为集合），与复 n 维空间等同看待，称为 n 阶复系数多项式空间. 例如，复 3 维空间中的点

$$(5-7.1i,\quad -3+\sqrt{2}i,\quad \pi-0.12i)$$

可以代表 3 阶复系数多项式

$$z^3+(5-7.1i)z^2+(-3+\sqrt{2}i)z+(\pi-0.12i),$$

而 3 阶复系数多项式

$$z^3+(-0.2-3.4i)z^2+(6.2-1990i)z+(\sqrt{7}+\sqrt{5}i)$$

也可以用复 3 维空间中的点

$$(-0.2-3.4i,\quad 6.2-1990i,\quad \sqrt{7}+\sqrt{5}i)$$

来表示. 在这样处理以后，利用代数几何和积分几何的知识，斯梅尔试图给对于 $z(0)=0$ 这个固定的初始点来说是坏的和较坏的多项式在整个多项式空间中所占的体积作一个估计.

高维空间的体积是怎么回事？实数轴是实 1 维空间，我们熟悉实数轴上的长度概念；实数 x-y 平面是实 2 维空间，我们熟悉平面上的面积概念；实数 x-y-z 空间是实 3 维空间，我们熟悉空间中的体积概念. 从 1 维 2 维 3 维推广到实 n 维空间，就建立了实 n 维体积的概念. 由于一个复数“等于”两个实数，即 $z=x+iy$ 这样一个复数可视同为 (x, y) 这样一对实数，所以复 n 维空间可视同为实 $2n$ 维空间.

整个空间的体积是无穷的，无穷量之间的比把握起来不易. 但在 n 阶复系数多项式空间 $\mathscr{P}$ 中，由 $|a_{n-1}|<r$，$|a_{n-2}|<r$，…，$|a_1|<r$，$|a_0|<r$ 所确定的复 n 维圆柱 $P(r)$ 的体积则是 $(\pi r^2)^n$ 这样一个有限数，容易把握得多. 这个复 n 维圆柱就是所有系数的绝对值都小于 r 的 n 阶复系数首一多项式的集合.

对于固定的参数牛顿迭代的初始点 $z(0)=0$，把与这个初始点匹配得非常不好（根本不收敛）和匹配得比较不好（收敛得

太慢）的多项式从这个复 n 维圆柱中刨去. 比较不好的多项式挖去越多，剩下的多项式与初始点 $z(0)=0$ 匹配得就越好，也就是说，从 $z(0)=0$ 开始的参数的牛顿迭代可以很快地进入这种多项式的快速收敛区，即很快找到原型牛顿迭代的良好初值.

若挖去部分的体积与复 n 维圆柱 $P(r)$ 的体积之比为 μ，$0<\mu<1$，那么如果对剩下来的所有多项式得到一个一致的结论 $\mathscr{A}$，我们也可以说结论 $\mathscr{A}$ 对任一 n 阶复系数多项式成立的概率至少是 $1-\mu$. 设想从全班 50 个学生中挑选了 4 个人，并发现剩下的 46 个同学的英文考试成绩都在 70 分以上，当然可以说该班任一同学"英文考试成绩在 70 分以上"这一结论成立的概率至少是 $1-4/50=92\%$. 所以，上述"结论 $\mathscr{A}$ 成立的概率至少是 $1-\mu$"的说法，自然并且合理.

正是这样的思路，引导斯梅尔得到他的下述主要定理：

定理 3 给定 n 和 $0<\mu<1$. 取换算参数 $\sigma=(\mu/150)^{3/2}/(n+2)^2$，则对复 n 维圆柱 $P(r)$ 中的多项式 $f(z)$，以下事实成立的概率至少是 $1-\mu$：只要适当选取步长参数 h，就可以保证从 $z(0)=0$ 开始的参数牛顿迭代在

$$s=4\left(3+\frac{8r\log(15/\sigma)}{\sigma^2}\right)^2$$

步之内达到 $f(z)$ 的一个良好初值.

定理的意义在于，对于在前面提出的统一从 $z(0)=0$ 出发仅靠调节步长参数 h 来利用参数牛顿迭代找良好初值的方案，证实这个方案在 s 步内成功的概率至少是 $1-\mu$.

前已说明，这个定理的证明要用到代数几何和积分几何的知识，不能在本书细述. 但我们应当捉摸一下定理的文字. 既然 $1-\mu$ 是结论成立的概率，那么 μ 就是允许结论失败的概率. 迭代步数 s 的表达式说明，μ 越小，σ 就越小（远远小于 1），而

s 也就越大. 这是很自然的：你要我得出一个至少对 40% 多项式成立的结论，我算出要迭代 1000 次；你要我得到一个至少对 70% 多项式成立的结论，我算出要迭代 10000 次. 反过来就是说，1000 次迭代就找到良好初值的要求较高，我只能保证 40% 的多项式能够成立；10000 次迭代才找到良好初值的要求就比较低，我可以保证 70% 的多项式能够成功.



图 3.5

定理中所需迭代步数 s 的表达式比较复杂，须先由 μ 和 n 得到 σ，再由 r 和 σ 算出 s，而且其中有对数关系. 对于 $r=1$ 的情形作耐心的换算，可以得出 $s\leqslant(100(n+2))^9/\mu^7$ 的不等式. 原来定理中用等号，保证在多少步内到达良好初值. 这个步数当然是上限，所以当时就可以改用“$\leqslant$”不等号. 现在在换算中做了放大，用“$\leqslant$”更自然. 因此，上述 $r=1$ 的换算，可以叙述为下面的推论：

推论 给定自然数 n 和实数 $0<\mu<1$. 对于系数 a_{n-1}，…，a_1，a_0 的绝对值都不超过 1 的 n 阶首一多项式

$$f(z)=z^n+a_{n-1}z^{n-1}+\cdots+a_1z+a_0,$$

只要步长参数选取适当，从 $z(0)=0$ 这个统一的初始点开始的参数牛顿迭代，可以在 $s=(100(n+2))^9/\mu^7$ 步内进入多项式的快速收敛区的概率，至少是 $1-\mu$.

不论是原型的牛顿迭代还是参数牛顿迭代，每做一次迭代

都要计算一次 $f(z)$ 的值和一次 $f'(z)$ 的值，其中 $f'(z)$ 是比 $f(z)$ 低一阶的多项式.笼而统之，我们可以说一次牛顿迭代相当于两次多项式计值.因此，如果你认为有必要，可以将定理 3 和推论中的迭代步数 s 乘 2，变成多项式计值次数.现在，$(100(n+2))^9/\mu^7$ 或它的 2 倍数的分子部分似已告诉我们，用参数牛顿迭代找良好初始所需要的迭代步数或多项式计值次数，是多项式阶数 n 的多项式增长函数. 从分子的表达式看，这个数算出来一般都很大，而找到良好初值以后再用原型的牛顿算法计算，就很快收敛到多项式的根（参看图 3.5）. 后面快速收敛的工作量与前面艰苦搜索良好初值的工作量相比，是微不足道的. 由于这个道理，我们可以把定理 3 或推论中的式子，看作是求根所需要的多项式计值总次数（你可以乘 2，不再赘述），不再只看作是寻求良好初始的工作量. 按照这样的理解，$(100(n+2))^9/\mu^7$ 的分子部分着实在暗示，牛顿算法是多项式时间算法.

对比头一章讲的多项式时间算法的概念，现在的结果多了分母 μ^7. 这个 μ 的出现，是数值计算复杂性理论的一个里程碑.

§4 从最坏情形分析到概率情形分析

本来，在数值计算的复杂性讨论的初期，只有当计算保证是收敛的时候，才能讨论一种算法的计算复杂性，特别是弄清楚它是多项式时间算法还是指数时间算法. 现在，对于计算有时根本不收敛的牛顿算法，斯梅尔却要讨论它的计算复杂性，这是一项巨大的挑战，它给计算复杂性理论带来一场深刻的富有成果的革命.

这一变革的标志，就是 μ 项的引入.

传统上，数学总是关注最坏情形．一个命题是否成立，就要看它是否对一切情形都成立．如果所有情形可以按照是否有利于得出命题的结论而区分出好坏及好坏的程度，那么只有当在最坏的情形时结论也成立，才能肯定命题为真；相反只要在最坏的情形时结论不能成立，就能论定命题不真．这就是数学上所讲的普遍性（generality）的含义．

近年来，数学开始关注“几乎肯定要发生”的事情．“几乎肯定要发生”是什么意思？请看一个简单的例子：拿一把刀朝实数轴砍去，几乎肯定要砍到无理数上．为什么这样说？请看下面的论证．

假定有一条长度为 1 的线段，当中有长度为 μ 的一小段涂了红颜色．当然，$0<\mu<1$．现在你闭起眼睛朝这个线段砍去，请问在砍中线段的条件下，你砍中红色小段的概率是多少？很自然，这个概率是 μ．

现在请问当你向 [0，1] 线段砍去的时候，砍中有理数的概率是多少？设这个概率是 μ，，我们要论证 $\mu=0$．大家知道，有理数是可数的，即可以编号为 t_1，t_2，t_3，…．如果用长度为 $\sigma\cdot 2^{-k}$ 的区间把 t_k 这个有理数盖住．那么盖住所有有理数的这无穷多个小区间的总长度为

$$\begin{aligned}&\sigma/2+\sigma/2^2+\sigma/2^3+\cdots\\&=\sigma(1/2+1/2^2+1/2^3+\cdots)\\&=\sigma(1/2)/(1-1/2)\\&=\sigma,\end{aligned}$$

注意括号内的数列是等比数列．由此可见，$\mu<\sigma$．这是因为那总长度为 σ 的无穷多个小区间已经把有理数全部盖住了，而砍中有理数的前提是砍中小区间．

注意上面的 σ 可以是任意正数．如取 $\sigma=10^{-1}$，说明 $\mu<$

10^{-4}；如取 $\sigma=10^{-5}$，说明 $\mu<10^{-5}$，…．但 μ 是一个固定的非负实数，它既然比任一正数都小，就必定是 0. 所以砍中有理数的概率为 0. 因此我们说，砍中有理数是**零概率事件**.

不是有理数的实数，就是无理数. 当你向 [0，1] 线段砍去时，砍不到有理数，就一定砍上了无理数. 所以，砍上无理数的概率为 1.

把 [0，1] 线段一段一段接起来，就得到实数轴. 由于可数个可数集合在一起仍是可数集，所以在实数轴上砍中有理数的概率仍然为 0，而砍中无理数的概率就是 1. 数学上，把发生一种事件的概率为 1 的事件，叫做概率 1 事件，或者**全概率事件**.

由此可见，发生的可能性（即概率）为百分之百的事件，有可能不发生；发生的可能性为 0 的事件，却仍有可能发生. 有人把这种认识表达为一个通俗的说法：0 不等于无. 这个说法是准确的. 同样，百分之百也不等于全体.

如果一个集合中的元素具有某种性质的概率是 1，我们就说集合中**几乎**所有元素都具有那种性质. 例如，我们可以说几乎每个实数都是无理数. 一个集合中几乎所有元素都具有某种性质，这就是近代数学上所讲的通有性（genericity）的含义.

从只承认普遍性质到也研究通有性质，现代数学取得了有声有色的进展，若干突破接踵而至. 这种情况，似未在我国的数学教育中得到应有的反映. 80 年代科学出版社出版的《英汉数学词汇》仍未留意区分 general 和 generic，使人感觉困惑. 简而言之，普遍性质是所有对象都具有的性质，而通有性质是百分之百的对象所具有但未必是所有对象都具有的性质.

现在，更进一步开始了概率性质的研究，不仅只研究全概率和零概率这两个极端，而且研究从 0 到 1 的中间概率. 就数

值计算的复杂性讨论而言，如果一种算法算得很好的概率是90%，剩下的10%之中有时算得不很好，有时算得很不好，要是拘泥于通有性的100%，就只能割爱不顾、把它归入不能得出好的结果的算法之中．这不仅在理论上十分可惜，而且对于指导实践也是很大的损失．具体地说，这一章讲的牛顿算法，下一章谈到的线性规划问题的单纯形算法，都是在实际计算中很受欢迎的方法．但是，这些算法表现出好的行为的概率都不是1．只有当计算复杂性理论开展了概率情形分析以后，人们才算对这些在实际计算中行之有效而大受欢迎的算法，有了一个科学的、合理的视角，不因它们有时是指数时间算法或甚至不收敛而简单地在计算复杂性理论中否定它们．反过来，倘若理论按照其标准判决一些经常十分有效的方法为不合标准的方法，把它们打入另册，那末标准的本身就令人怀疑，理论也就站不住脚．

这就是斯梅尔引进 μ 项，把计算复杂性理论推进到可以进行概率情形分析的意义．

斯梅尔说，它是在动力系统的框架内研究算法及其计算复杂性问题的．这很值得玩味．

粗略地说，动力系统（dynamical systems）理论是关于微分流形上向量场的流的理论，欧氏空间是最简单的微分流形．给定一个向量场，就是给定一组微分方程．向量场的流，就是微分方程的积分轨线．如果不是进行连续观察，而是对流进行离散采样，就得到离散的动力系统．我们可以用下面的比喻，说明连续轨线和离散采样的关系：萤火虫的飞行轨迹是连续的．但是当萤火虫在黑暗之中飞行时，只有当萤火虫一闪一闪发光时才能观察到它的位置．我们所做的观察，就是对萤火虫的飞行轨迹进行离散采样．

如果对多项式 $f(z)$ 进行牛顿迭代，那末迭代公式 $z_{k+1} = \varphi(z_k)$，$k=0, 1, 2, \cdots$就是一个分子分母都是多项式的所谓有理函数. 给定 $f(z)$，也就给定了有理的迭代函数 $\varphi(z)$. 以平面上任一点为 z_0，只要避开使迭代函数的分母 $f'(z)$ 为 0 的顶多 $n-1$ 个点，都可以 z_0，z_1，z_2，…这样一直迭代下去. 所以,牛顿迭代用到每个具体的多项式上,都确定复数 z 平面上的一个离散动力系统.

这时我们进一步设想，在复数 z 平面上随机地选取点作为初值来进行牛顿迭代. 如果从某点开始的迭代收敛，就在这个点处打一个黑点，如果从某点开始的迭代不收敛，就不打黑点. 这样一点一点进行试验，就会得到白纸黑点的一个图形. 用计算机判别迭代是否收敛，只能是设定一个迭代次数，如果迭代了这么多次还看不出收敛，就判为不收敛. 所以，上面所说的黑点都位于牛顿迭代的快速收敛区（不一定是斯梅尔在定理 1 中所确定的那种快速收敛区）. 再设想按照收敛速度的不同层次划分，这些快速收敛区本身的构造如何？扩张的方式如何？当采用参数的牛顿迭代时，步长参数的变化对收敛速度的影响如何，对收敛区的构造的影响如何？凡此种种，都可以说是动力系统方式的设想. 这些设想的准确刻划和深入讨论，仍然是有待研究的问题.

§5 算法之比较和配合

现在，我们试比较一下第二章关于库恩算法的成本估计和这一章斯梅尔关于牛顿算法的成本估计.

两种算法对付的都是多项式求根的问题. 第二章的定理 3

说，对于任一所有系数的绝对值均不超过1的 n 阶多项式，采用库恩算法，按照误差不超过 ϵ 的要求算出多项式的全部 n 个根所需要的多项式计值次数，不超过 $100n^3\log_2(n/\epsilon)$。这样，算出一个根所需要的多项式计值次数平均不超过 $100n^2\log_2(n/\epsilon)$。

第三章定理3的推论则说，对于任一所有系数的绝对值均不超过1的 n 阶多项式，以下事实成立的概率至少是 $1-\mu$：只要适当选取步长参数 h，就可以保证从 $z(0)=0$ 这个统一的初始点开始的参数牛顿迭代在 $(100(n+2))^9/\mu^7$ 步内为多项式求根的牛顿算法找到一个良好初值。

$100n^2\log_2(n/\epsilon)$ 次多项式计值对 $(100(n+2))^9/\mu^7$ 次牛顿迭代，已经提供了一个很好的比较。特别是从估计式中可以清楚地看出成本估计随多项式阶数 n 增长而增长的速率。它们都是多项式时间算法。

略嫌不足的是参数不完全一致，估计库恩算法的成本用 n 和 ϵ，而估计牛顿算法的成本用 n 和 μ。要使参数完全一致，无非两种途径。一是让牛顿迁就库恩，用参数牛顿迭代找到一个良好初值以后，再把达到精度要求 ϵ 所需要的那部分成本加上去。这样做，牛顿成本势必增加，而遇上按牛顿方法看来是坏的多项式，沿斯梅尔设计的路线（图3.5）将永远到达不了快速收敛区，进一步的迭代更无从谈起。所以这一途径较难进行。

另一种途径就是让库恩迁就牛顿。这是我们应当做的，何况库恩算法比牛顿算法晚了差不多三百年。

这样，就要考虑用库恩算法为多项式寻求牛顿算法的良好初值的问题。本来，良好初值这个概念对于在任何情况下总会成功的库恩算法来说完全是多余的，但为了在成本估计之间做一个参数完全一样的无懈可击的比较，我们用库恩算法来解决进入牛顿算法所要求的快速收敛区的问题。

这样做当然有所损失．例如，牛顿算法所冀求的快速收敛区只能对没有重根的多项式建立．对于有重根的多项式 $f(z)$，其表征值 $\rho(f)=0$，这时斯梅尔按 $|f(z_0)|<\rho(f)/13$ 确定的快速收敛区根本不存在．所以，尽管用库恩算法可以任意接近每一个根，却因为根本没有快速收敛区而被判为未进入快速收敛区．

因此，只好不顾 $\rho(f)=0$ 和 $\rho(f)$ 太小的情况．对于剩下的 $\rho(f)$ 不太小的每个多项式 $f(z)$，设存在一个常数 $\rho_0>0$ 使得 $\rho(f)\geqslant\rho_0>0$ 对这些多项式总是成立．按照斯梅尔的充分条件，只要 $|f(z_0)|<\rho(f)/13$ 就是良好初值了．既然库恩算法可以任意接近准确根，何愁在一定时刻后不会进入快速收敛区？按照这个想法进行从 μ 到 ϵ 的换算，（为什么上述想法实际上是从 μ 到 ϵ 的换算，请读者思考．）我们可以得到下面的定理：

定理　对于所有系数的绝对值均不超过 1 的 n 阶首一多项式，以下事实成立的概率至少是 $1-\mu$：用库恩算法顶多经过

$$200(n+2)^3\log_2(n/\mu)$$

次多项式计值，就可以为多项式找到一个牛顿算法所需要的良好初值．

现在，$200(n+2)^3\log_2(n/\mu)$ 次多项式计值和 $(100(n+2))^9/\mu^7$ 次牛顿迭代，这两个估计式就是参数完全一样的成本估计了．

斯梅尔研究牛顿算法的计算复杂性的论文，题为《代数基本定理与复杂性理论》，发表在 1981 年的《美国数学会通报》上．这是数值计算复杂性理论从只做最坏情形分析到也做概率情形分析的开创性论文．在这篇论文中，斯梅尔还提出讨论其它算法的计算复杂性的一系列课题．可以说，正是斯梅尔的这篇论文，吸引本书作者与徐森林和库恩教授一起，完成了本书第二章介绍

的研究和这一节介绍的比较定理.

斯梅尔教授开设的研究生课程,经常邀请相关领域的专家报告自己的成果. 1983 年春天,他以动力系统框架和计算复杂性理论为主题向研究生开课. 若干校外的和国外的教授也每课必到,不放过学习和讨论的机会. 斯梅尔教授邀请本书作者报告库恩算法的计算复杂性讨论,作为 83 年春季课程的第一个外邀讲演.

作为成本估计,$100n^3\log_2(n/\varepsilon)$ 和 $200(n+2)^3\log_2(n/\mu)$ 当然比 $(100(n+2))^9/\mu^7$ 好得多. 但是决不能由此而低估斯梅尔的工作的重要意义,因为我们讨论的是具有良好几何拓扑结构的库恩算法,而斯梅尔对付的是难以驾驭的牛顿算法. 就牛顿算法的坏的情况来说,收敛性都谈不上,更何况成本估计. 斯梅尔果断而巧妙地排除了坏的和较坏的情况,得出一个出色的概率估

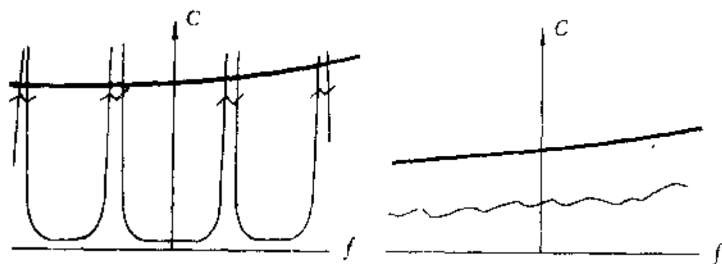


图 3.6

计. 他的工作是开创性的. 当时我画了一张理论成果评价示意图,横轴 f 表示多项式,纵轴 C 表示成本. 图 3.6 的左图,细实曲线表示用牛顿算法求多项式根的实际成本,粗实曲线表示斯梅尔对牛顿算法成本的估计. 图 3.6 的右图,细实曲线表示用库恩算法求多项式根的实际成本,粗实曲线表示我们对库恩算法

成本的估计.这是一张笼统的无标度的示意图,但是清楚地表明我们是在做一项相对容易的工作,所以得到的估计式比较好(粗实曲线比较低).斯梅尔的出色之处,在于图中小折线表示的截断处理,正如前面强调过的,这是真正开创性的工作.

这张用粉笔当时画在黑板上的评价图,得到在座的教授和研究生的赞赏。

当然,开创性的工作也并不一定已经十全十美.从构造性数学的观点来看,斯梅尔设计的用参数牛顿迭代寻找良好初值的方法在彻底的意义上说还不是完全构造性的.这是美中不足的地方.问题在于步长参数 h 的选取.不错,斯梅尔给出了 h 的表达式,h 可以由 ζ 和 $\mathscr{K}$ 确定,但是 ζ 和 $\mathscr{K}$ 却又依赖于 $\rho(f)$,$\rho(f)$是 $f(z)$的临界值的绝对值的最小者,而 $f(z)$的临界点就是 $f'(z)$的根.这样,为了构造性地解决寻找多项式 $f(z)$的良好初值的问题,必须先解决寻求另一个多项式 $f'(z)$的根的问题.这样的环环相扣的构造性,在逻辑上有点讲不过去.

指出美中不足,并不等于我们已经可以做得更完美.我们估计,那会是更加困难得多的问题.

从构造性数学的观点来看,库恩算法却总是可以在由精度要求 ε 和多项式阶数 n 预先完全确定的多少次多项式计值之内,把多项式的 n 个根一无例外全找出来.所以,无论库恩算法的本身还是我们对于库恩算法计算复杂性的讨论,都完全是构造性的.

库恩算法的优点是保险成功,牛顿算法的长处是进入快速收敛区后收敛极快.看来,两种算法配合使用,可以把二者的长处结合起来.事实上,采用库恩算法先把根的位置大致确定,然后改用牛顿算法迅速向准确根靠近,这在理论分析和实际计算中都是很有希望的方案.

第四章　线性规划问题算法的竞争

20 世纪末叶，世界进入计算机时代.

计算机的用途很广，从账目管理、文件编译、办公室自动化，一直到机器人服务，解放了多少人力！

但是，以上提及的计算机的应用，基本上不属于科学计算方面. 要问计算机对付得最多的科学计算问题是什么，线性规划问题的求解可算一个. 有资料说，当今世界经济效益最大的科学计算方法，就是线性规划问题的单纯形算法.

这一章主要介绍围绕线性规划问题若干算法的竞争，竞争的标准就是它们的计算复杂性. 与多项式求根问题相比，线性规划问题算法的讨论更受关注，也更加困难. 许多地方，我们将限于定性的说明，并且会穿插有关的科学故事. 科学史上许多有声有色的故事，对后人的启迪决不在发现和发明本身之下. 可惜国内的著述出版，常爱把故事放在不屑一顾的地位.

§1 线性规划问题

美国加利福尼亚理工学院教授乔尔·富兰克林(J. Franklin)为大学生写的《数理经济学方法》,是我所钟爱的课本.该书1980年由权威的斯普林格—瓦纳格出版社出版.数理方面的大学课本,通常总是由概念、演绎、结论这样一些环节组成,或者再加一点应用.富兰克林这本书的显著特点,则是在作者个人学问故事的"切切私语"中展开数理经济学的有关内容.在大学数理课本中,行文如此亲切、秀丽的,实在是凤毛麟角.

《数理经济学方法》主要论述线性规划、非线性规划和不动点定理.在正文的首页,作者回顾1958年的调研旅行,纽约美孚石油公司负责接待他们的专家,竟是他当年的同窗好友艾伯特博士,他的办公室"看起来像橄榄球场那么大".富兰克林写道:

> "艾,你成了世界大人物了."我说.
>
> "啊,没有的事,…,确实不是."他说.
>
> 他还是像从前那样腼腆,但景况已大不相同.在纽约大学时,艾和我曾经在没有空调的满是灰尘的小办公室里做博士后研究工作.终究因为是学者,我们被预料将是清寒的.不知为什么,艾却没有应验这一预料.
>
> 我们对计算机的作用作了很好的谈话.公司有一个巨大的计算中心,我知道它花费了几百万美元.那是50年代的几百万美元哪!果真值得吗?我问道:
>
> "艾,我知道石油公司很有钱.但总没有人愿意浪费它.公司要花多少时间,才能收回对计算中心的投资?"
>
> 他想了一会儿,显然进行了粗略的心算,然后说:

“大约两个星期.”

“真是不可思议.”我说,“你们用计算机做什么问题?”

“主要是线性规划.”

艾作了详细说明.应用线性规划,他们能够对生产和销售作出最优决策,给公司和顾客都带来很大的经济利益.

现在具体谈谈什么是线性规划.先看几个例子.

例1 投资经营.

假设银行有100亿元,分别用作贷款(L)和证券(S).贷款赚得较高的利息.证券所赚利息较低,但具有流通的方便:它们随时可按市场价格出售.

设贷款利率10%,证券的利率平均5%,则总盈利为$0.10L+0.05S$.银行希望总盈利最大.

如果没有约束条件,这个问题没什么可讨论的,令$L=100$即可.事实上是有各种约束:

流动性约束.

例如银行要求流动性资金保持在25%或更多,即

$S \geqslant 0.25(L+S)$,或

$L-3S \leqslant 0$.

贷款额约束.

银行有一些信用极好的老主顾,必须保证对他们的贷款.估计这个额度是30亿元,就有

$L \geqslant 30$.

另外,当然还有总资金约束

$L+S \leqslant 100$

和符号约束

$L \geqslant 0, \qquad S \geqslant 0$.

把$z=0.10L+0.05S$叫做目标函数,那么整个问题可以表述为

$$\begin{aligned}
\max \quad & z=0.10L+0.05S, \\
\text{s.t.} \quad & L-3S\leqslant 0, \\
& L\geqslant 30, \\
& L+S\leqslant 100, \\
& L\geqslant 0,\quad S\geqslant 0.
\end{aligned}$$

这里,max 是 maximize 之缩写,max z 就是使 z 达到最大;s.t. 是 subject to 的缩写,表示"受约束于".

例 2 饮食问题.

1945 年,斯蒂格勒(G. Stigler)发表论文讨论这样的问题:营养合理的饮食的最小费用是多少?

假设共有 n 种可用的食物,消费量分别为 $x_1,x_2,\cdots,x_n$,n 种食物的价格分别为 $p_1,p_2,\cdots,p_n$.

设供分析的营养(包括热量、各种维生素、蛋白质、脂肪、酶、粗纤维素、氨基酸、矿物质等等)共有 m 种,每种的需要量至少是 b_i, $i=1,\cdots,m$. 又设每单位第 j 种食物所含有营养成分 i 的量是 a_{ij}, $i=1,\cdots,m$, $j=1,\cdots,n$. 那么,饮食问题可以表述为

$$\begin{aligned}
\min \quad & z=p_1x_1+\cdots+p_nx_n, \\
\text{s.t.} \quad & a_{i1}x_1+\cdots+a_{in}x_n\geqslant b_i,\ i=1,\cdots,m, \\
& x_1\geqslant 0,\cdots,x_n\geqslant 0.
\end{aligned}$$

其中,min 是 minimize 的缩写,min z 表示要使 z 达到最小.

请读者务必明白上述表述的含意. 由于从这个饮食问题开始的对线性规划问题和其他问题的研究,斯蒂格勒获得 1982 年度的诺贝尔经济学奖.

例 3 资源配置

假设石油公司用其总量分别为 $s_1,\cdots,s_m$ 的 m 种粗制品提炼价格分别为 $p_1,\cdots,p_n$ 的 n 种精制品. 怎样进行配置,才能使收入达到最大?

设生产一单位精制品 j 需要粗制品 i 的量为 a_{ij}, $i=1,\cdots,$

$m, j=1, \cdots, n$,这样,问题可表述为

$$\max z=p_1x_1+\cdots+p_nx_n,$$
$$\text{s.t. } a_{i1}x_1+\cdots+a_{in}x_n\leqslant s_i, \quad i=1,\cdots,m, \ x_1\geqslant 0,\cdots,x_n\geqslant 0.$$

许多应用数学问题和现实经济问题,都具有和例 3 一样的数学形式. 我们把这样的问题称为**线性规划问题**. 这种数学形式的基本特点是:**目标函数**和**约束条件**都是线性式子. 如果目标函数和 m 个约束条件之中有一个非线性式子,就叫做非线性规划,本书只谈线性规划.

这里要注意的是,问题可以是目标函数求最大的 $\max z$,例如收入、利润,也可以是目标函数求最小的 $\min z$,例如成本、费用. 另外,约束条件中的不等号,既可以是"≤",也可以是"≥",甚至可以是等号.

上面概括的,已经是线性规划问题的一般形式. 数学研究的一种重要方法,就是从特殊到一般. 首先研究一个特例. 从具体的特例开始,比较容易入手. 特例弄清楚了,往往也就有了解决一般问题的办法. 下面,我们就研究例 4,它是例 3 的一种特殊情形,是变量数目 $n=2$,约束条件数目 $m=5$ 这样的容易把握的特殊情形. 我们仍然强调几何地分析问题和解决问题.

例 4 某工厂利用 3 种原料生产 2 种产品. 产品对原料的需求如下表,表中并列出原料的存量和产品的价格. 目标是使收入达到最大.

原　　料	产品 1	产品 2	原料总数
1	5	9	4500
2	1	4	1600
3	4	1	2400
产品价格	10	7	

按照上述已知条件，问题可表达为

$$
\begin{aligned}
\max \quad & z=10x_1+7x_2,\\
\text{s.t.} \quad & 5x_1+9x_2\leqslant 4500,\\
& x_1+4x_2\leqslant 1600,\\
& 4x_1+x_2\leqslant 2400,\\
& x_1\geqslant 0,\quad x_2\geqslant 0.
\end{aligned}
$$

符合所有约束条件的点的集合，叫做线性规划问题的**可行区域**. 可行区域中的每一个点，亦即每一个符合所有约束条件的点，叫做问题的一个**可行解**. 使得目标函数达到最大(max 问题)或最小(min 问题)的可行解，叫做**最优可行解**. 线性规划问题的目标，或者说目的，就是求出问题的最优可行解.

例 4 有 5 个约束条件，把它们在 x_1-x_2 平面上画下来，就得到图 4.1 粗实线所围的可行区域.

图中有一组平行细实线，它们是目标函数的**等值线**，同一线上各点的目标函数值相等. 沿着这组等值线，右上方向是目标函数值增加的方向. 可行区域的右上方向有两个角 A 和 B. 按照等值线和可行区域边界线的斜率关系，易知可行区域中目标函数值最大的点是 B，所以 B 就是问题的最优可行解. 至此，例 4 这个具体的线性规划问题，已得到完全解决.

例 4 虽然简单，却已经体现了线性规划问题的要义. 我们强调首先把简单的特殊的例子弄懂. 如果你真的懂了，你就能想象图 4.2 所表示的三个线性规划问题的内容，并且知道如果这些线性规划问题有解，解一定出现在 A、B 或 C、D 或线段 EF 上的一点. 象 D 这样结果全是 0 的解叫做**平凡解**，通常没有多大意思. 但平凡解确实有出现的可能.

图 4.1 和图 4.2 的 4 个线性规划问题的最优可行解，都出现在可行区域的角上. 下面我们要说明，这是必然的结局.

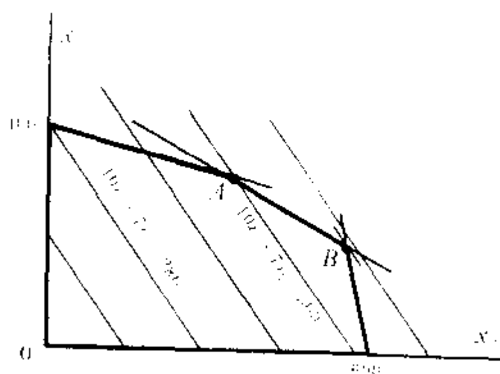


图 4.1

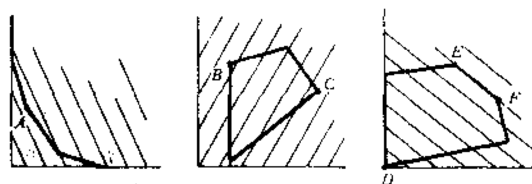


图 4.2

首先，我们需要一点凸集的概念.欧氏空间中的一个集合，如果它的任意两点的连线段上的每一点仍属于这个集合，那么它就是一个**凸集**.例如，图 4.3 曲线或折线围成的 4 个图形（集合），都是凸集，而图 4.4 曲线或折线围成的 4 个图形，都不是凸集.但是要注意，如果把图 4.3 的 4 个图形的所有点放在一起作为一个集合，则这个集合不是凸集.

凸集概念也用于**无界集合**，即以原点为中心的不管半径多么大的球（圆）都不能把它包在里面的集合.例如，图 4.2 左边的

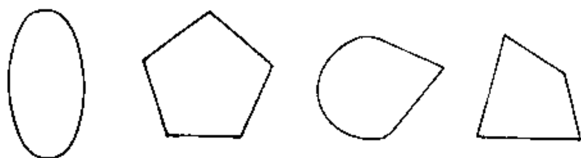


图 4.3



图 4.4

线性规划问题的可行区域就是一个无界集合. 特别要注意的是：平面的每个象限都是无界的凸集，欧氏空间的每个卦限都是无界的凸集，全空间是凸集.

凸集有一个十分重要的性质：一刀切下去，变成两半，那末每一半都是凸集，而且切口也是凸集. 这里必须注意的是：切，就要切透，切到底，不能切到半途就停下来，否则你很容易把正五边形切成五角星，把凸集切成不是凸集. 当然，一定要用平直的刀（“线性”）.

线性规划问题都是在受约束于若干个约束条件的情况下使目标函数达到最大或最小，约束条件和目标函数都是线性的. 符合所有约束条件的点的集合，就是问题的可行区域. 变量 x_1，…，x_n 的 n 维空间，是一个凸集. 每一个约束条件，相当于切一刀. 如果这个约束条件是不等式，就相当于一刀切下去取符合不等式的那一半. 如果这个约束条件是等式，就相当于一刀切下去只取切口留下来. 可见，对于一个凸集，做一次这样的“切、取”操

作，得到的仍然是一个凸集. 线性规划问题的可行区域就是由 n 维空间这个大凸集一刀一刀切下来所得到的区域，所以必定是一个凸集.

在线性规划问题中，变量 $x_1,\cdots,x_n$ 的个数 n，叫做问题的**维数**. 因为不等式约束是“切一刀取一半”的操作，所以只有不等式约束的线性规划问题的可行区域的维数，通常和问题的维数一样. 但是等式约束却是“切一刀只取切口”的操作，所以带有等式约束的线性规划问题的可行区域的维数，一定比问题的维数低.

例 5 线性规划问题

$$\begin{aligned}
\min\quad & z=x_1+2x_2+1991x_3,\\
\text{s.t.}\quad & 6x_1+2x_2+15x_3=300,\\
& -x_1+x_2\leqslant 10,\\
& x_1\geqslant 0,\ x_2\geqslant 0,\ x_3\geqslant 0
\end{aligned}$$

的维数是 3. 由于有一个等式约束，可行区域的维数是 $3-1=2$，如图 4.5 黑框所围的平面四边形.

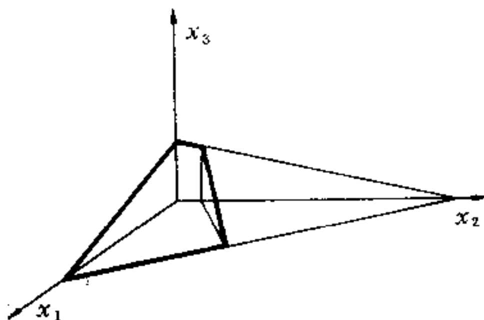


图 4.5

不管维数是否降低，重要的是肯定，线性规划问题的可行区

域一定是凸集.问题的解就是在可行区域中找一点,使得目标函数值达到最大或最小.目标函数是线性的.读者可以几何地想象,在可行区域上面有一块平板,平板的高度代表目标函数的值,那么,不管平板斜得厉害与否,最高点和最低点一定对应着可行区域的边界上的点.几何想象,是数学思考的重要方法.相信读者已能自己想象上面所说的几何图象,不必我们在这里画一幅图代替你的想象.

线性规划问题的可行区域不仅是一般的凸集,而且一定是一种称为**凸多面体**的凸集.事实上,欧氏空间用上面所说的切法一刀一刀切下去,最后得到的几何对象,就是凸多面体.作为2维的凸多面体,就是凸多边形.作为3维的凸多面体,读者可以想象为一颗宝石,它的每个端面都是平的凸多边形.凸多面体有许多**顶点**.上面已经说明,线性规划的最优可行解一定出现在可行区域的边界上.如果你对那个几何想象式的"证明"确实掌握的话,那么现在按照同样的几何想象,知道**最优可行解一定可以在顶点上达到**.

这里要注意两点:首先,并非所有线性规划问题都一定有最优可行解.例如图4.2左的问题,如果等值线的值向右上方增加并且问题是求最大值的问题,那么就没有最优可行解.更有一些线性规划问题的可行区域是空集,也就是没有可行解,那就更谈不上最优可行解(读者现已应不难想象这种情况).上面说的是,如果有最优可行解,那么最优可行解一定可以在顶点上达到.其次,上面的结论并不排斥在非顶点的地方也给出最优可行解的可能.例如图4.2右的问题,如果目标函数值向右上方增加并且问题是求最大值的问题,那么线段 EF 上的每一点都给出问题的最优可行解.但是不管怎样,只要最优可行解是存在的,就一定可以在一个顶点上达到最优可行解,例如 E 或 F.

综上所述，线性规划问题的可行区域如果非空，就必是凸多面体；最优可行解如果存在，就一定可在可行区域凸多面体的顶点上达到.

§2 丹齐克的单纯形算法

从上一节的分析我们知道，线性规划问题的最优可行解如果存在，就一定可以在可行区域多面体的某个顶点上达到. 由此可知，如果把可行区域这个凸多面体每个顶点上的目标函数值都算出来并加以比较，就可以解决线性规划问题.

线性规划问题，都可以化成以下的**标准形式**：

$$\begin{aligned}
\max\quad & z=c_1x_1+\cdots+c_nx_n,\\
\text{s.t.}\quad & a_{i1}x_1+\cdots+a_{in}x_n=s_i,\quad i=1,\cdots,m,\\
& x_1\geqslant 0,\cdots,x_n\geqslant 0.
\end{aligned}$$

标准形式的特点是，除符号约束外其余约束都是等式约束. 例如上节例 4 的问题

$$\begin{aligned}
\max\quad & z=10x_1+7x_2,\\
\text{s.t.}\quad & 5x_1+9x_2\leqslant 4500,\\
& x_1+4x_2\leqslant 1600,\\
& 4x_1+x_2\leqslant 2400,\\
& x_1\geqslant 0,\ x_2\geqslant 0,
\end{aligned}$$

引入 3 个**松弛变量** x_3,x_4,x_5，就可以写成

$$\begin{aligned}
\max\quad & z=10x_1+7x_2,\\
\text{s.t.}\quad & 5x_1+9x_2+x_3=4500,\\
& x_1+4x_2+x_4=1600,\\
& 4x_1+x_2+x_5=2400,
\end{aligned}$$

$$x_1, x_2, x_3, x_4, x_5 \geqslant 0.$$

这已是线性规划问题的标准形式，读者容易看出它和原来的写法完全等价.

标准形式的另一个要求是：等式约束的右端 $s_1, \cdots, s_m$ 全是非负数. 这是很容易做到的，必要时全式乘 -1 就可以了. 至于 $\max z$ 还是 $\min z$ 是无所谓的. 有些书规定只用 $\min z$，那么遇上 $\max z$ 时，把原来的 $z = c_1x_1 + \cdots + c_nx_n$ 乘一个 -1 作为新的 z，就变成 $\min z$ 问题.

标准形式线性规划问题变量的数目 n，称为问题的**维数**（以后我们所说的维数，都指这个维数），等式约束的数目 m，称为问题的**阶数**. 当然，我们要求 $n \geqslant m$.

研究表明，一个标准形式的线性规划问题的可行区域，通常是 n 维空间中的一个 $n-m$ 维凸多面体，它最多可有

$$C_n^m = \frac{n!}{m!\ (n-m)!}$$

个顶点. 最粗糙的计算

$$C_n^m = \frac{m+1}{1} \cdot \frac{m+2}{2} \cdots\cdots \frac{n}{n-m} > \left(\frac{n}{n-m}\right)^{n-m} = \left(1 + \frac{m}{n-m}\right)^{n-m}$$

已经告诉我们，C_n^m 是 n 的指数函数. 所以，如果我们采用**枚举法**，一个顶点一个顶点那样计算目标函数值然后加以比较，这样一种枚举算法的时间复杂性函数就一定是 n 的指数函数.

早在 20 世纪 30 年代，苏联数学家康托洛维奇（Leonid Kantorovich）从研究资源配置入手，写成了《生产组织和计划中的数学方法》一书. 这是关于线性规划问题的最早的著作. 可惜，由于学术政策方面的障碍，康托洛维奇的工作在当时未曾得到应有的评价，国外也不知道.

1947 年，美国数学家丹齐克（George B. Dantzig）提出了线性规划问题的一般模型和求解线性规划问题的单纯形算法. 从此，线性规划理论和应用得到了广泛发展.

二次大战期间，丹齐克就在美国空军的一个小组从事资源分配和计划编制的工作. 大战以后，他回到伯克利加州大学，并取得了博士学位. 这时，他成为美国空军审计长的数学顾问，从事计划工作机械化的研究.

作为一个数学家，首先要把问题表达清楚，这就是所谓建立模型的工作. 这时，美国劳工统计局所做的列昂节夫（W. Leontief）投入产出模型使他受到很大启发. 列昂节夫是十月革命后移居美国的俄裔经济学家，他的投入产出模型是矩阵结构的一种线性模型，在概念上非常简单同时又足够精细，对实际制订计划很有帮助. 当时，经济学家们已经形成了他们的线性规划模型，并且已经有了一些求解的方法. 现在看来，这些方法都很蹩脚.

1947 年初，丹齐克拜访了经济学家柯普曼（T. J. Koopmans），从他那里知道线性规划问题还没有一种有效的、一般的算法. 这年夏天，丹齐克就研究从可行区域凸多面体的一个顶点出发，沿着凸多面体的棱，走向目标函数值更优的下一个顶点的方法.

p 维凸多面体的每个端面，都是一个 $p-1$ 维的凸多面体. 例如 3 维凸多面体（“钻石”）的端面都是 2 维的凸多面体（凸多边形）.

空间中仿射无关的 $p+1$ 个点的凸包，称为 p 维**单纯形**. 1 维单纯形是线段，2 维单纯形是三角形，3 维单纯形是四面体. p 维凸多面体可以分割为一个个 p 维单纯形，$p-1$ 维凸多面体可以分割为一个个 $p-1$ 维单纯形. 这样分割时，可以不出现新

的顶点．这就是说，单纯形的顶点都是原凸多面体的顶点．

凸多面体这样分割为一个个单纯形以后，属于同一单纯形的两个顶点称为是相邻的顶点．丹齐克的单纯形算法，通过一种矩阵表格的运算，提供了从一个顶点走向相邻顶点中目标函数值最优的顶点的方法．

翻开任何一本介绍单纯形算法的教科书，并且花一点耐性把它读下去，你就知道学会单纯形算法的操作并不是一件困难的事．不过，这需要列出许多表格，占用许多篇幅，所以本书只好割爱，而代之以上面这样比较笼统的介绍．

提出单纯形算法以后，丹齐克有两个不放心．首先，线性规划的一般理论当时还不完备．为此，他到普林斯顿高等研究院，向人们称为当代最伟大的数学家的冯・诺意曼(J. von Neumann）请教．原来，当时冯・诺意曼刚刚和摩根斯滕写完《对策论和经济行为》这本数理经济学的划时代的巨著．这本巨著以稍许不同的方式，为线性规划理论奠定了坚实的基础．关于线性不等式的理论，关于凸多面体的理论，就是关于线性规划的理论．

另一方面，丹齐克耽心单纯形算法在实际计算时是不是有效，会不会算得很慢．大约一年以后，在 1948 年六月，空军小组的成员告诉丹齐克，单纯形算法对于所有试验过的问题，都非常有效．这真出乎算法发明者本人的预料．40 年以后，丹齐克这样回顾自己当时的感受：

> 解一个阶数为 m 的线性规划问题大体上只需要 $2m$ 到 $3m$ 次迭代，这个结果确实令人吃惊．我确实未预料到结果会这么了不起．我当时还没有解高维问题的经验，我不能依赖自己的几何直觉．例如，我的直觉告诉我，从一个顶点或许需要经过很多步才能移动

到相邻的下一个顶点，而实际上只需要几步. 简单地说，在高维空间，人们的直觉一文不值.

线性规划问题的理论价值和经济价值，使它在经济学研究方面也占有重要的地位. 自从1969年颁发经济学诺贝尔奖以来，上面提到的康托洛维奇，柯普曼和斯蒂格勒都曾因为与线性规划问题有关的工作而获奖，列昂节夫则因为发展了投入产出方法而获奖. 冯·诺意曼健在的时候，还没有经济学诺贝尔奖，不然的话，他更是当之无愧.

现在，线性规划可说是理论上完备，方法上可靠. 这里所讲的方法，就是丹齐克提出的实际使用效果很好的单纯形算法.

关于线性规划理论的发展，还有一个值得回味的故事. 在向冯·诺意曼请教以后不久，丹齐克参加了美国计量经济学会在威斯康星举行的一次学术会议. 会议的参加者中有霍特林(H. Hotelling)，柯普曼斯，冯·诺意曼这样的大人物，也有许多刚开始从事学术研究的年轻人，他们后来也都相当知名. 他第一次向这样一批有许多著名学者在内的听众演讲关于线性规划的概念，心情相当紧张. 事实上，那时候丹齐克完全是个无名小辈.

丹齐克讲完以后，会议主席照例请大家进行讨论. 在冷场一会儿以后，著名经济学家霍特林举手发言. 他站起来，带着人们熟悉的微笑，毫不客气地说：

"但是，我们大家都知道，世界是非线性的." 讲完，就庄严地坐了下来.

丹齐克正不知道怎样回答大人物霍特林的质疑，冯·诺意曼举起了手. 他说："主席先生，如果演讲人不介意的话，我将乐于代他回答这个问题." 丹齐克当然同意. 冯·诺意曼对大家说：

"报告人把他的题目叫做线性规划，非常认真地叙述了他的公理. 如果你有什么应用问题是满足这些公理的，你可以采用他的方法. 如果你没有这样的应用问题，那当然不必勉强."

这真是一个科学哲学的问题. 从根本上说，霍特林的看法完全正确，世界确实是非线性的，而且是高度非线性的. 但是这种看法，即使是完全正确的看法，不应当捆住人类解决面临问题的手脚. 可幸的是，线性系统（线性不等式组或线性方程组）还是可以用于近似地解决人们在实际规划问题中遇到的绝大部分非线性关系，而且解决得十分成功. 本章提到的投入产出经济分析和线性规划的理论和方法，都是极富哲理启发的例子.

§3 哈奇安的椭球算法

成熟的单纯形算法在1951年公布. 因为它总是可以很快找到一个线性规划问题的最优可行解，或者可以很快判断一个线性规划问题没有最优可行解，所以很受应用部门欢迎，迅速被普遍使用起来. 这种局面的技术背景，是数字电子计算机的发展和普及.

但是，直到80年代以前，单纯形算法的成功，还缺乏计算复杂性理论方面的基础. 50年代，丹齐克本人就在一篇文章中表示，他相信单纯形算法的计算成本，是线性规划问题规模 n 和 m 的线性函数. 换句话说，他相信单纯形算法是多项式时间算法中最好的一种算法——线性时间算法. 但是，他从千千万万实际计算情况中总结出来的上述猜想，一直未能从理论上得到

证明.

1972 年，美国数学家克里（V. Klee）和闵蒂（G. Minty）在《不等式》杂志上发表的一篇题为《单纯形算法是不是很好?》的论文中，构造了一类线性规划问题，如果用单纯形算法来解这类问题，几乎要检验可行区域凸多面体所有顶点上的目标函数值，然后才能找出最优可行解. 尽管克里和明蒂人为构造的那类线性规划问题在实际应用中从未遇到过，但是根据数学论证总是着眼于最坏情形分析的传统视角，单纯形算法就不是理论上企求的多项式时间算法. 一方面，在实际应用中单纯形算法总是计算得很好，另一方面，就最坏情形而言它又是一种指数时间算法. 从 70 年代到 80 年代初期，单纯形算法一直承受着实际使用总是很好而理论分类却很差的巨大矛盾.

这么好的算法却还不是多项式时间算法，这就引起了一个更深刻的问题：矛盾是来自解决线性规划问题的单纯形算法这种具体算法，还是来自线性规划问题的本身? 换句话说，对于线性规划问题，究竟有没有一种多项式时间算法? 要知道，有些问题，例如第一章的梵天宝塔移位问题，之所以没有多项式时间算法，是问题本身的难度所决定的，算法设计得再好，也不可能是多项式时间算法.

随着计算机应用和计算机科学的发展，单纯形算法在实算经验和理论成果之间的强烈反差，越来越引起人们关注. 整个 70 年代，线性规划问题究竟有没有多项式时间算法，成了关注的中心. 正是在这种情况下，苏联青年学者哈奇安（L. G. Khachian）发表了一篇题为《线性规划的多项式时间算法》的论文，提出了线性规划问题的椭球算法.

椭球算法专门对付形如

$$\begin{cases} a_{11}x_1+\cdots+a_{1n}x_n<b_1, \\ \cdots \\ a_{m1}x_1+\cdots+a_{mn}x_n<b_m, \end{cases}$$

这样的整系数线性不等式组，所有不等号都是严格不等号．符合所有不等式的点的集合，就是问题的可行区域．与前面所说的那些线性规划问题不同的是，现在只要找到可行区域中的一个点，即只要找到符合所有不等式的一个点，问题就算解决了．不难证明（本书从略），线性规划问题都可以归结为上述形式的严格不等式组的问题，求线性规划问题的最优可行解，就归结为求上述严格不等式组的可行区域中的任意一点．

椭球算法有若干不同的形式．现在，我们介绍某种条件下应用的椭球算法的基本思想．为叙述方便，我们约定，当 E_k 表示形如

$$\frac{x_1^2}{a_1^2}+\cdots+\frac{x_n^2}{a_n^2}\leqslant 1$$

的椭球体时，e_k 表示相应的形如

$$\frac{x_1^2}{a_1^2}+\cdots+\frac{x_n^2}{a_n^2}=1$$

的椭球面．注意 E_k 的维数总是 n，而 e_k 的维数总是 $n-1$．当 $a_1=\cdots=a_n$ 时，椭球就变成了圆球．

首先，从 n 和 m 可以算出正数 $r=r(n,m)$ 和 $v=v(n,m)$，使得只要问题有解，那么以原点为中心 r 为半径的球体 E_0 与严格不等式组的可行区域凸多面体之交集 P 非空，并且交集 P 的体积至少是 v．

算法：

步 0　取 E_0 为以原点为中心 r 为半径的球体，置 $k:=0$．

步 1　如果椭球 E_k 的中心符合严格不等式组，那么这个中心就是所求的解，停机．否则，E_k 的中心至少违反一个不等式，

不妨设违反

$$a_{i1}x_1+\cdots+a_{in}x_n<b_i,$$

那么

$$a_{i1}x_1+\cdots+a_{in}x_n=b_i$$

所决定的 $n-1$ 维超平面 l_{k+1} 与椭球面 e_k 交于一个 $n-2$ 维的椭球面 b_{k+1}. 在 P 的另一侧作超平面 l'_{k+1} 与 l_{k+1} 平行，并且切椭球面 e_k 于点 A_{k+1}. 过点 A_{k+1} 和 $n-2$ 维椭球面 b_{k+1} 作一个 $n-1$ 维椭球面 e_{k+1}，把以它为边界的椭球体记作 E_{k+1}. 置 $k:=k+1$，回到步 1.

图 4.6 是 $n=2$ 时算法的示意图，这时粗实线所围的凸多边形就是问题的可行区域，它与 E_0 的交集 P 如阴影所示. 由于 $n=2$，算法中的 $n-1$ 维超平面就是 1 维的直线，$n-2$ 维椭球面就是两个点.

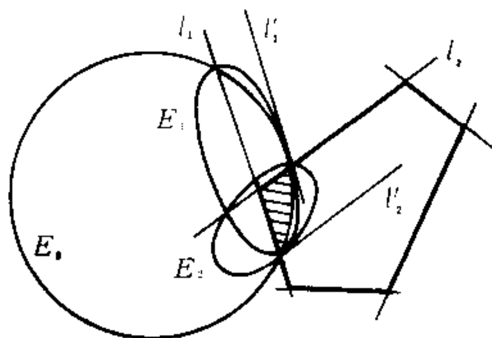


图 4.6

可以证明，存在一个由 n 和 m 决定的正数 $c=c(n,m)$ 满足

$0<c<1$，使得用椭球算法依次得到的椭球 E_0，E_1，E_2，…，符合

$$E_{k+1}\text{的体积} \leqslant c \cdot E_k \text{的体积}$$

的关系，由此即得估计式

$$E_k \text{的体积} \leqslant c^k \cdot E_0 \text{的体积}，k=1，2，3，\cdots$$

另一方面，从算法的叙述易知，每次得到的新的椭球 E_{k+1}，都仍然把集合 P 整个包在里面．所以，算法产生的每个椭球的体积，都比 v 大．由此可见，一定有一个正整数 K，使得产生 E_0，E_1，…，E_K 之后，计算就会中止，否则会和上述体积的估计式矛盾．

依据算法，计算停下来是因为已经找到了问题的解．但是要注意算法前讲的"只要问题有解，集合 P 的体积至少是 v"的语句：任何算法不能排除原问题没有解的可能性．所以，椭球算法是这样一种算法：如果问题有解，一定可以在 K 步之内找到问题的一个解；如果计算了 K 步仍未找到问题的解，就可以断定原问题根本没有解．

无论找到具体的解还是断定没有解，都不必再找下去，问题都已彻底解决．所以，上面的叙述，已经确立了椭球算法的收敛性．

更为重要的是：经过细心的演算，这个 K 可以用 n 和 m 这两个变元的多项式表示出来．换句话说，随着 n 和 m 的增长，K 按照 n 和 m 的多项式的关系增长．所以，椭球算法是一种多项式时间算法．

前面说过，求解线性规划问题的最优可行解，都可以归结为求相应的严格不等式组的可行解，而后者可以用椭球算法"在多项式时间内"得到解决．这就说明，线性规划问题有多项式时间算法．

读者在前面已经知道线性规划问题的重要性．哈奇安的论文，使人们舒了一口气．这是 1979 年震动数学界、计算机科学界和许多应用部门的大事．有趣的是，当哈奇安一举扬名，美国记者去访问他的老师时，老师却回答“想不到”．事实上，哈奇安的同学和同事们几年来针对组合的或非光滑的或凸的最优化问题，已经对椭球算法进行了深入的研究，取得了很好的成果．哈奇安的功绩，可以说“只是”别出心裁地把椭球算法用到线性规划问题上去．嫁接就是创造，综合就是创造．这也是哈奇安给我们的启示．

细心的读者会发现，在上述椭球算法中，“过点 A_{k+1}和 $n-2$ 维椭球面 b_{k+1}作 $n-1$ 维椭球面 e_{k+1}”，是一件可能相当困难或相当麻烦的事．但是哈奇安们早先已经研究出了一个公式，可以很容易完成这项工作．还要说明的是：标准的椭球算法原来要求超平面 l_{k+1}经过椭球 E_k 的中心，但这毫不影响我们所作的论证．

§4　卡马卡的内点算法

哈奇安的论文发表以后，人们对椭球算法寄予厚望．可是，在应用问题的实际计算中，椭球算法老是比单纯形算法算得慢，而且慢很多．于是，又出现了新的反差：理论上，单纯形算法是指数时间算法，椭球算法是多项式时间算法，所以椭球算法比单纯形算法优越；实践中，椭球算法却远远不是单纯形算法的对手．随后的几年时间里，人们致力于改进椭球算法，但收效甚微，改变不了椭球算法在实际计算中远远落在后边的局面．

1984 年，旅美印度青年数学家卡马卡（N. Karmarkar）发

表论文，提出线性规划问题又一种新算法，这种新算法不仅在理论上是多项式时间算法，而且据说有时算得比单纯形算法快.特别对于求解某些大型的线性规划问题，据说新算法比单纯形算法快好几倍. 卡马卡的论文，再一次引起轰动. 人们设想，新算法果真对若干大型问题计算得比单纯形算法快的话，就有可能使得一些原来由于规模太大而无法解决的线性规划问题得到有效的解决.

单纯形算法的特点是，从可行区域凸多面体的一个顶点开始，沿着多面体边沿上的棱，一个顶点一个顶点走下去，每走一步，情况都得到改善（目标函数值变小或变大），直至得到问题的最优可行解，或者判定问题没有最优可行解.

对于一个凸多面体来说，如果从这头走到那一头，规定只许沿着边沿上的棱走的话，这似乎是一种绕圈子的走法. 从这头到那头，能不能从凸多面体的内部插过去呢？卡马卡提出的这种新算法正好具有从内部插过去的性质，经常要计算的点，是可行区域凸多面体的内点. 所以，卡马卡提出的新算法，以及人们受卡马卡的启发陆续改进和形成的一大类具体算法，统称线性规划问题的**内点算法**.

卡马卡算法因为要反复做投影变换和仿射变换，技术性很强，所以很难在这本小册子里作类似于对椭球算法那样的介绍. 也许，下述笼而统之的描述，对某些读者仍有少许帮助. 设多面体为 P，a 是 P 的一个内点. 通过变换，把它们变为多面体 P' 和点 a'，使得 a' 是 P' 的“中心”. 这时，局限在 P' 中以 a' 为球心的某个球面上解原来的线性规划问题，设所得为 b'. 最后，把 b' 通过上述变换的逆变换送回到 P 内去，记作 b. 再以 b 作为新的 a，重复上述算法.

卡马卡算法不仅理论上是多项式算法，而且有时确实算得

比单纯形算法快，这是它吸引人的地方．由于线性规划问题在理论上和在效益上的重要性，内点算法一时成为国际数学界关注的新发展．1986 年召开的四年一度的国际数学家大会，邀请卡马卡做最高规格的一小时报告．

对于以卡马卡算法为代表的内点算法的研究，有一种现象，就是别人发表的论文比卡马卡及其同事们所写的更多．原因之一，是因为卡马卡及其同事是美国贝尔实验室的研究人员．贝尔实验室是美国一个很有实力的多科性的研究和开发集团．出于保护知识产权的考虑，卡马卡不能发表他的算法的若干细节．这就使得别人采用卡马卡算法计算同一个线性规划问题时，效率往往不如卡马卡及其同事宣称的那么高．这种差距积累久了，就会引起一定程度的怀疑．的确，在卡马卡头一篇论文发表大约 4 年以后，曾经发生过他在重要的国际学术会议上及时透露机器程序的一些细节以平息疑虑的情况．

今天，如果有人说卡马卡算法已经把椭球算法远远抛在后边，那基本上确是如此．但是，哈奇安的椭球算法首次证明了线性规划问题有多项式时间算法，功不可没．至于卡马卡算法和单纯形算法的比赛，却仍在进行之中．两种算法在理论方面之长短，上面谈过一些，下一节还要谈．在实际计算方面，比较普遍同意的看法是：单纯形算法仍然是最受欢迎的具有很高效率的算法，但在某些情况下卡马卡算法有它自己的优势．

不管情况如何发展，卡马卡算法开创的内点算法的发展，是 80 和 90 年代应用数学研究领域的盛事．这是一场相得益彰的“竞争”．

§5 斯梅尔论证了丹齐克的信念

前已说过，丹齐克在1947年提出了线性规划问题的单纯形算法. 当时，他在理论和实算两个方面都不是很有把握的. 经过一年的试验，人们告诉他，单纯形算法的计算效率意外地好. 在理论方面，他把线性规划理论系统化，总结多年的实践经验，完善了各种专门技巧，在1963年出版了题为《线性规划及其推广》的大部头著作.

在那个时候，明确的数值计算复杂性理论尚未形成. 但是，算法效率如何，当然已是人们关心的问题. 单纯形算法的效率如何呢？丹齐克未能从理论上完整地回答这个问题，却在书中明白地提出了自己的信念：

> “人们相信，在约束数 m 固定的情况下，对一个随机选取的线性规划问题来说，用单纯形算法解决这个问题所需要的迭代次数，正比于变量数目 n.”

为了更准确地理解这段话，我们采用矩阵符号来表达标准形式的线性规划问题. 设问题是

$$\begin{aligned}
\min\quad & z=c_1x_1+\cdots+c_nx_n, \\
\text{s.t.}\quad & a_{11}x_1+\cdots+a_{1n}x_n\geqslant b_1, \\
& \vdots \\
& a_{m1}x_1+\cdots+a_{mn}x_n\geqslant b_m, \\
& x_1,\ \cdots,\ x_n\geqslant 0.
\end{aligned}$$

令矩阵

$$A=\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix},$$

向量

$$b=\begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}, \quad c=\begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix}$$

同时，变量亦表示成

$$x=\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix},$$

那么，原问题可以紧凑地用一句话写下来：

找满足 $x\geqslant 0$，$Ax\geqslant b$ 的 $x\in \boldsymbol{R}^n$ 使 $z=c\cdot x$ 达到最小.

$x\in \boldsymbol{R}^n$，表示 x 是 n 维向量. 同样，我们知道 $A\in \boldsymbol{R}^{mn}$，$b\in \boldsymbol{R}^m$，$c\in \boldsymbol{R}^n$. 两个同维向量作点乘，就是分量依次相乘然后求和，具体来说，$c\cdot x=c_1x_1+\cdots+c_nx_n$.

按照上述标准形式，我们只要知道

线性规划问题的数据 (A, b, c)，

整个问题也就清楚了. 大家可以看到，矩阵方式的数据表示十分简洁. 这里注意，向量也看成矩阵，是只有一列的矩阵. 反过来，矩阵亦可看成向量. 例如上面的 A，是 mn 维向量，所以我们写 $A\in \boldsymbol{R}^{mn}$.

这样合起来，(A, b, c) 就是一个 $N=mn+m+n$ 维向量，或者说是 $\boldsymbol{R}^N$ 中的一个点. 于是，在给定 m 和 n 的条件下，令 $N=mn+m+n$，那么任何一个 m 阶 n 维的线性规划问题都可以表示为 $\boldsymbol{R}^N$ 中的一个点；反过来，$\boldsymbol{R}^N$ 中任何一个点也都表示一个 m 阶 n 维的线性规划问题.

基于这样的认识，所谓随机选取的线性规划问题，就是 $\boldsymbol{R}^N$

中随机选取的点.

1982 年，斯梅尔证实了丹齐克的信念. 当然，局限于最坏情形分析的传统，是不会得到这样的结果的，因为克里和闵蒂已经说明，在最坏的情况下，单纯形算法是指数时间算法. 斯梅尔在 $\boldsymbol{R}^N$ 空间中引进概率测度，进行概率情形分析. 他证明了，在约束数 m 固定的情况下，线性规划问题单纯形算法的计算成本（以迭代次数或顶点转换次数衡量）的平均数，随变量数目 n 线性增长.

概率情形分析的基本模式是：除去概率为 μ 的最坏的情形，对剩余的 $1-\mu$ 那么多的问题，结果将是如何？平均情形分析，则要把最坏情形也考虑在内. 比如说一个问题有 10 种情形，发生概率都一样. 最坏情形的运算量是 100，而其余 9 种情形的运算量都只是 2，那么运算量的平均数将是 12. 读者可以想象，由于也要对付最坏情形，平均情形分析是一项难度更大的工作.

在这里，我们不准备介绍斯梅尔的整个做法，而只提出他的指导思想. 斯梅尔写道：

> “牛顿算法和分片线性算法都可以看作路径跟踪算法. 这是我们几篇论文的主题. 用这种方法，你可以把丹齐克的单纯形算法也看作路径跟踪算法. 因此，线性规划的单纯形算法和牛顿算法之间的联系并不令人感到意外. 事实上，在线性互补问题的框架中，这种关系确实是非常简单和非常自然的.”

我们只说一说什么是线性互补问题，什么是路径跟踪算法. 了解当今热门的这两个论题的概貌，是会有好处的.

线性互补问题 (linear complementarity problem) 的一般提法是：给定 $(\mathscr{A}, q)$，其中 $\mathscr{A}$ 是 $M\times M$ 矩阵，q 是 M 维向量，求 M 维向量 x 使得 $\Phi(x)=q$，这里 $\Phi(x)=x^{+}+\mathscr{A}x^{-}$，其中

x^+是向量 x 把负分量都置 0 后所得的向量，x^-是向量 x 把正分量都置 0 后所得的向量.

给定线性规划问题（A，b，c），令

$$\mathscr{A}=\begin{bmatrix}0 & -\boldsymbol{A}^T\\ \boldsymbol{A} & 0\end{bmatrix},\quad q=\begin{bmatrix}c\\ -b\end{bmatrix},$$

就得到相应的线性互补问题（$\mathscr{A}$，q），其维数关系是 $M=m+n$. 可以证明，线性规划问题（A，b，c）有一个解当且仅当相应的线性互补问题（$\mathscr{A}$，q）有一个解. 并且它们的解自然地相互对应.

许多应用数学问题，特别是运筹学方面提出来的问题，都可以这样转化为线性互补问题.

现在说说路径跟踪方法.

为求映射 f：$\boldsymbol{R}^n\to\boldsymbol{R}^n$ 的零点（根），我们选择一个零点清楚的辅助映射 g：$\boldsymbol{R}^n\to\boldsymbol{R}^n$，按照

$$H(t,x)=tg(x)+(1-t)f(x)$$

或其它方式，构造连接 g 和 f 的同伦

$$H:[0,1]\times\boldsymbol{R}^n\to\boldsymbol{R}^n.$$

当参数 $t=1$ 时，$H(1,x)$ 就是 $g(x)$；当参数 $t=0$ 时，$H(0,x)$ 就是 $f(x)$. 在一定的条件下，同伦 H 的零点集

$$H^{-1}(0)=\{(t,x)\in[0,1]\times\boldsymbol{R}^n:H(t,x)=0\}$$

是一些互不相交的光滑的简单曲线，一头是 g 的零点，一头是 f 的零点. 从已知的 g 的零点出发，沿着这些曲线走，就可以到达待求的 f 的零点. 这就是**同伦方法**，一种典型的**路径跟踪算法**的原始思想.

如果进一步对空间 $[0,1]\times\boldsymbol{R}^n$ 进行分割成一个个 $n+1$ 维单纯形的所谓单纯剖分，并且取

$$\Psi:[0,1]\times\boldsymbol{R}^n\to\boldsymbol{R}^n$$

在每个顶点上和 H 一致，在各单纯形内部是仿射映射，那么 Ψ 就是分片线性映射，其零点集

$$\Psi^{-1}(0)=\{(t,x)\in[0,1]\times R^n:\Psi(t,x)=0\}$$

在一定条件下是简单折线. 沿着这些折线走的方法，叫做**分片线性同伦算法**.

图 4.7 之左右，分别是同伦算法和分片线性同伦算法的示意图，后者也叫**单纯同伦算法**.

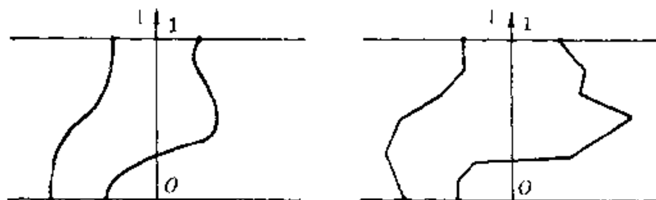


图 4.7

在我国，线性互补问题和路径跟踪算法，都是有待开展的研究工作，愿更多同行关心它.

§6 复杂性讨论的学科环境

研究数值计算，当然要考虑计算成本或算法效率的问题. 在这个意义上，数值计算的复杂性问题，可以追溯到很早以前. 20 世纪中叶以来，由于计算机技术的发展，许多以前无法对付的问题变得可以考虑，计算复杂性的讨论开始迅速发展. 然而，直到 20 世纪 70 年代，这种讨论都带有局部的和渐近的特征.

斯梅尔的参加，是计算复杂性讨论得到强大推动和取得丰硕成果的转机. 以往的计算复杂性讨论，大量是计算速度和收

敛阶的讨论，例如可以这样提出问题：在（通常关于初始值的）什么样的条件下，计算有几阶敛速．这就是局部的和渐近的讨论．斯梅尔带来的转折，则是从整体上而不是从局部探讨某种算法的总的计算成本或者平均计算成本，回答在随机地选取一个初始值或者随机地选取一组问题参数的条件下，解决某种类型的数值计算问题的平均成本是多少．

新的讨论需要新的学科环境．这一节作为全书的结尾，对此作一粗线条的介绍．

拓扑学 研究数值计算的总成本或者平均成本，并且是从整体上考虑问题，就必须引进拓扑学和几何学．

在某种意义上，拓扑学是数学讨论从局部走向整体的桥梁．每个局部都等同于欧氏空间开集的几何对象，就是所谓流形．所以，流形是由欧氏开球拼接起来的．如果拼接得光滑，就得到光滑流形或微分流形．流形是局部定义的整体概念．关于向量场积分曲线的讨论，如果限于欧氏空间，就是常微分方程的内容，如果在流形上整体考虑，就属于动力系统范畴．这就是一个典型的说明．值得注意的是，在流形上展开的数学讨论，往往只需要形式上局部的验证，即可得到整体的结论．

拓扑学在其它领域的应用发展很快．之所以会这样，并不是由于代数拓扑或微分拓扑的高深理论或最新成果，而主要是因为拓扑学本质上整体的讨论方式适应了其它领域的要求，是因为拓扑学的一些基本方法（如同伦，单纯剖分，论定值域几乎每一点都是光滑映射的正则值的萨德定理等）在其它领域开拓了应用．通晓线性代数和多元微积分基础的读者，不难掌握拓扑学的这些最基本但也是最重要的内容．

当然，这并不是说只用到拓扑学的基础的概念和浅层的结果．例如，在斯梅尔关于算法的“拓扑复杂度”的研究之中，用

到了代数拓扑中关于“辫群”的上同调环的结果，而且有些成果不能从单复变情形推广到多复变情形，恰恰是因为代数拓扑中一些问题未获得解决.

代数几何 古典代数几何是在仿射空间和射影空间研究多项式方程组的解集（代数簇）的构造的理论. 所以，代数几何本来就是和方程理论紧密联系的. 不仅许多算法原来就是对多项式方程组提出来的，而且多种通用算法的复杂性讨论都必须首先从多项式对象的情形开始. 这个开端虽然已经取得可喜成果，但是还远远谈不上结束. 代数几何对复杂性讨论的重要性由此可见. 许多关于多项式的讨论都以齐次方程的讨论为基础，这就需要在射影空间中展开. 代数几何的前置课程是线性代数，再加上一点仿射几何，射影几何，拓扑学和复分析.

几何概率 几何概率是非概率数学工作者容易接受的数学概念. 向一块条纹板随机地发射，击中深色条纹的概率是多少？这就是几何概率的例子.

对于牛顿迭代，考虑初始值对收敛性或效率的影响. 在确定了好与坏的标准以后，好的初始值在勒贝格（Lebesgue）测度的意义上占全空间的比率，就是在空间中任取一点作为初始值算法运行得好的概率. 反过来，固定一个初始值对不同的多项式或不同的函数进行牛顿迭代，如果能够在相应的函数空间中引进适当的测度，也可以讨论从这个固定的初始值开始，对任一函数进行牛顿迭代时算法运行得好的概率是多少.

积分几何中以关于不同维数体积之间的关系的富比尼（Fubini）定理为中心的有关内容，是复杂性理论中几何概率讨论的基本手段.

单叶函数理论 一元情形的牛顿迭代是在复平面上进行的. 收敛性是对于算法的最基本的考虑. 因为在单根（单零点）

附近，多项式或解析函数的行为都近似于单叶函数，所以复杂性讨论就自然联系到复变函数论中的单叶函数理论.具体地说，与比勃巴赫猜想——德布兰吉斯定理有关的工作，都在斯梅尔等人的复杂性讨论中起着重要作用（参看第三章第2节）. 拓扑学和几何学的整体处理，加上单叶函数理论的细致分析，促成了许多成果的结晶.

与国外的学者比较，我国一些数学家在拓扑和代数方面略逊，却以分析的功夫见长. 对这种格局，也建议采取一种参与的途径，使数学家们得以发挥自己之所长.

本书讲了斯梅尔的开创性工作，并且一再提及数值计算的老法宝——牛顿算法. 我们并不隐晦这个着重点. 斯梅尔在1986年世界数学家大会上的报告中说：

> “任何算法如果已被验证能够解决非线性方程组问题，那么它必定是牛顿型算法.”

这句话，值得我们去体味.